(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 December 2001 (20.12.2001)

PCT

(10) International Publication Number
WO 01/97447 A2

(51) International Patent Classification⁷: H04L 12/00

(21) International Application Number: PCT/SE01/01301

(22) International Filing Date: 7 June 2001 (07.06.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/210,761 12 June 2000 (12.06.2000) US
09/709,643 13 November 2000 (13.11.2000) US

(71) Applicant: TELEFONAKTIEBOLAGET LM ERICSSON (publ) [SE/SE]; S-126 25 Stockholm (SE).

(72) Inventor: RUNE, Johan; Terrängvägen 5, S-181 30 Lidingö (SE).

(74) Agent: GULLSTRAND, Malin; Ericsson Radio Systems AB, Patent Unit Research, S-164 80 Stockholm (SE).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 01/97447 A2

(54) Title: RANDOM IDENTITY MANAGEMENT IN SCATTERNETS

(57) Abstract: In an ad-hoc network, network identities are assigned to each network. Using the network identities, a node can determine whether a neighbor node (not connected to the node) is a member of the same network as the node or whether a neighbor node (not connected to the node) is a member of the same network as another neighbor node (not connected to the node). When two networks merge the network identity of one of them is selected and broadcast throughout the part of the merged network that has to change its network identity. To avoid multiple networks which were once members of the same network, from having the same network identity, network identities are periodically reassigned and broadcast throughout each network. Further, when a connection between two nodes is broken the nodes can determine whether they are still members of the same network before initiating a change of network identity procedure.

RANDOM IDENTITY MANAGEMENT IN SCATTERNETS

BACKGROUND

The present invention relates to ad-hoc networks. More particularly, the present invention relates to forming ad-hoc networks.

Conventional networking protocols are based on the characteristics and/or features of fixed networks. In fixed networks, the network configuration typically does not change. Although nodes can be added and removed in fixed networks, the route traveled by data packets between two nodes typically does not change. The disadvantage is that fixed networks cannot be easily reconfigured to account for increases in data traffic, also called system loading. Accordingly, when system loading increases for one node, the surrounding nodes are likely to experience increased delays in the transmission and reception of data.

In contrast to fixed networks, ad-hoc networks are dynamic. An ad-hoc network is formed when a number of nodes decide to join together to form a network. Since nodes in ad-hoc networks operate as both hosts and routers, ad-hoc networks do not require the infrastructure required by fixed networks. Accordingly, ad-hoc networking protocols are based upon the assumption that nodes may not always be located at the same physical location.

Bluetooth is an exemplary ad-hoc networking technology. Bluetooth is an open specification for wireless communication of both voice and data. It is based on a short-range, universal radio link, and it provides a mechanism to form small ad-hoc groupings of connected devices, without a fixed network infrastructure, including such devices as printers, PDAs, desktop computers, FAX machines, keyboards, joysticks, telephones or virtually any digital device. Bluetooth operates in the unlicenced 2.4 GHz Industrial-Scientific-Medical (ISM) band.

FIG. 1 illustrates a Bluetooth piconet. A piconet is a collection of digital devices, such as any of those mentioned above, connected using Bluetooth technology in an ad-hoc fashion. A piconet is initially formed with two connected devices, herein referred to as Bluetooth devices. A piconet can include up to eight Bluetooth devices. In each piconet, for example piconet 100, there exists one master Bluetooth unit and one or more slave Bluetooth units. In FIG. 1 Bluetooth unit 101 is a master unit and unit 102 is a Bluetooth slave unit.

According to Bluetooth technology a slave unit can only communicate directly with a master unit. FIG. 2 illustrates a piconet with a master unit 201 and a plurality of slave units 202-208 arranged in a star network topology. If slave unit 202 wishes to communicate with slave unit 206, slave unit 202 would have to transmit the information it wished to communicate to master unit 201. Master unit 201 would then transmit the information to slave unit 206.

A scatternet is formed by multiple independent and unsynchronized piconets. FIG. 3 illustrates an exemplary scatternet 300. In FIG. 3, piconet 1 includes a master node 303 and the slave nodes 301, 302 and 304; piconet 2 includes a master node 305 and the slave nodes 304, 306, 307 and 308; and piconet 3 includes a master node 309 and the slave nodes 308, 310 and 311. To implement a scatternet it is necessary to use nodes which are members of more than one piconet. Such nodes are herein referred to as forwarding nodes. If, for example, node 301 wishes to communicate with node 310, then nodes 304 and 308 might act as forwarding nodes by forwarding information between the two piconets and in particular between nodes 301 and 310. For example, node 301 transfers the information to the master node of piconet 1 node 303. Master node 303 transmits the information to forwarding node 304. Forwarding node 304 then forwards the information to master node 305, which in turn, transmits the information to forwarding node 308. Forwarding node 308 forwards the information to master node 309 which transmits the information to the destination node 310. Furthermore, it should be pointed out that a single piconet is just a trivial case of the more general scatternet concept. Hence, in this document the term "scatternet" refers to either a single piconet or multiple interconnected piconets.

Each Bluetooth unit has a globally unique 48 bit IEEE 802 address. This address, called the Bluetooth Device Address (BD_ADDR) is assigned when the Bluetooth unit is manufactured and it has never changed. In addition, the Master of a piconet assigns a local active member address (AM_ADDR) to each active member of the piconet. The AM_ADDR, which is only three bits long, is dynamically assigned and deassigned and is unique only within a single piconet. The master uses the AM_ADDR when polling a slave in a piconet. However, when the slave, triggered by a packet from the master addressed with the slave's AM_ADDR, transmits a packet to the master, the slave includes its own AM_ADDR (not the masters) in the packet header.

Even though all data is transmitted in packets, the packets can carry both synchronous data, on Synchronous Connection Oriented (SCO) links which is mainly intended for voice traffic, and asynchronous data, on asynchronous connectionless links (ACL) links. Depending on the type of packet that is used, an acknowledgment and retransmission scheme is used (not for SCO packets transferring synchronous data) to ensure reliable data transfer, as well as forward error correction (FEC) in the form of channel coding.

FIG. 4 illustrates a conventional Bluetooth packet. The conventional Bluetooth packet consists of access code 410, header 420 and payload 430. The header 420 contains the AM_ADDR followed by some control parameters, e.g., a bit indicating acknowledgment or retransmission request of the previous packet, when applicable, and a header error check (HEC). The access code 410 in the packet can be of three different types including a channel access code (CAC), a device access code (DAC) or an inquiry access code (IAC).

The channel access code identifies a channel that is used in a particular piconet, i.e., in essence the channel access code identifies the piconet. Accordingly, all packets exchanged within a piconet carry the same channel access code. The channel access code is derived from the BD_ADDR of the master unit of the piconet. The device access code is derived from a BD_ADDR of a particular Bluetooth unit. The device access code is used for special signaling procedures, e.g., the PAGE procedure. There are two types of inquiry access codes, the general inquiry access code (GIAC) and the dedicated inquiry access code (DIAC). Both the general inquiry access code and the dedicated inquiry access code are used in the INQUIRY procedure.

The format of payload 430 depends on the type of packet. The payload of an ACL packet consists of a header, a data field and a cyclic redundancy check (CRC). However, AUX1 packets do not include a CRC and the payload of an SCO packet consists only of a data field. In addition, there are hybrid packets including two data fields, one for synchronous data and one for asynchronous data. Packets in which the payload does not include a CRC are neither acknowledged nor retransmitted.

Since ad-hoc networks are dynamic, ad-hoc networking technology typically has a neighbor discovery feature. The neighbor discovery feature allows one node to find any other node with which the first node can communicate, and consequently form an ad-hoc network with. A neighbor discovery procedure in Bluetooth is known as the INQUIRY procedure. Once a Bluetooth unit knows of neighboring nodes, a Bluetooth unit can connect to the neighboring node using the PAGE procedure.

FIG. 5 illustrates the signaling performed between two Bluetooth units for neighbor discovery and connection establishment. A Bluetooth unit, such as Bluetooth unit 1, wishing to discover neighboring nodes broadcasts an INQUIRY message. The Bluetooth unit then waits and listens for an INQUIRY RESPONSE message. An INQUIRY message consists of only an inquiry access code. The inquiry access code can be a general inquiry access code, which is sent to discover any Bluetooth unit in the neighborhood, or a dedicated inquiry access code, which is sent to discover only certain types of Bluetooth units, for which the particular dedicated inquiry access code is dedicated.

When a neighboring node, such as Bluetooth unit 2, receives an INQUIRY message, the neighboring node will respond with an INQUIRY RESPONSE message. The INQUIRY RESPONSE message is really an FHS (Frequency Hop Synchronization) packet. A FHS packet is illustrated in FIG. 6. The FHS packet includes fields for parity bits, lower address part (LAP), Scan Repetition (SR), Scan Period (SP), upper address part (UAP), non-significant address part (NAP), class of device, AM_ADDR, internal value of the units clock (CLK), and Page Scan Mode. The LAP, UAP and NAP together comprise the BD_ADDR. The SR, SP and Page Scan Mode fields are control parameters used during the page procedure. The AM_ADDR field can be used to assign an AM_ADDR to a Bluetooth unit which is becoming a slave in a piconet, i.e., the AM_ADDR field is used only when the FHS packet is used in the PAGE procedure, and the undefined field is reserved for future use.

An FHS packet is also used for other purposes in a Bluetooth system, e.g., for synchronization of the frequency hop channel sequence. By listening for INQUIRY RESPONSE messages the Bluetooth unit that initiated the INQUIRY procedure, e.g., Bluetooth unit 1, can collect the BD_ADDR and internal clock values, both of which are included in the FHS packet, of the neighboring Bluetooth units.

When a Bluetooth unit desires to establish a connection with a neighboring node the Bluetooth unit sends a PAGE message. A PAGE message consists of the Device Access Code (DAC) which is derived from the BD_ADDR of the paged Bluetooth unit. A Bluetooth unit, e.g., Bluetooth unit 2, receiving a PAGE message including its own DAC responds with an identical packet, i.e., a packet including only the DAC of the paged Bluetooth unit. The paging Bluetooth unit, i.e., Bluetooth unit 1, then replies with an FHS packet, including the BD_ADDR of the paging Bluetooth unit (Bluetooth unit 1), the current value of the internal clock of the paging Bluetooth unit (Bluetooth unit 1), the AM_ADDR assigned to the paged Bluetooth unit (Bluetooth unit 2) and other parameters. The paged Bluetooth unit (Bluetooth unit 2) then responds with its DAC and thereby the connection between the two Bluetooth units is established.

If the paging Bluetooth unit is already the master of a piconet, the paged Bluetooth unit has now joined this piconet as a new slave unit. Otherwise, the two Bluetooth units have just formed a new piconet with the paging Bluetooth unit as the master unit. Since the INQUIRY message does not include any information about its sender, in particular not its BD_ADDR, the Bluetooth unit that initiated the INQUIRY procedure is the only one that can initiate a subsequent PAGE procedure. Thus, the Bluetooth unit initiating an INQUIRY procedure will also be the master of any piconet that is formed as a result of a subsequent PAGE procedure. However, if considered necessary, the roles of master and slave can be switched using a master-slave-switch mechanism. This, however, can be a complex and extensive procedure, potentially resulting in a redefinition of the entire piconet, involving all other slave units in the piconet.

The INQUIRY and PAGE procedures are well defined in the current Bluetooth standard. These are all the tools that are needed to form a new Bluetooth piconet or join an existing one. However, even though the tools are well specified, there are no rules or guidelines as to how to use them. When neighbors are discovered there is no way to know who to connect to in order to form an appropriate piconet. Further, using a master-slave-switch mechanism is an extensive procedure and it is difficult to determine when to use it in order to improve the efficiency of a piconet. Hence, piconets will more or less form at random, often resulting in far from optimal piconet and scatternet structures. Further, the Bluetooth units forming a piconet or a scatternet will not have an idea as to the structure and interconnection of various piconets.

Piconet and scatternet forming procedures would be facilitated, and more efficient piconet and scatternet topologies would be possible to achieve, if more information about the involved Bluetooth units could be exchanged before the piconets and scatternets are actually established. Accordingly, it would be desirable to provide pieces of information during the INQUIRY and PAGE procedures which aid a Bluetooth unit in determining an efficient method for scatternet forming.

Further, the INQUIRY RESPONSE messages only include information about the responding node itself. Accordingly, even if an inquiring node would collect INQUIRY RESPONSE messages from all the nodes in its vicinity, it would not obtain any information about how these nodes are interconnected, which nodes would be reachable through each other, and which nodes are connected to the same scatternet. Having a picture of how the different neighboring nodes are interconnected via "islands of connectivity" in the form of separate scatternets, would be valuable information for a node trying to establish connectivity, especially while trying to establish a so-called Maximum Connectivity Scatternet (MCS). For more information regarding Maximum Connectivity Scatternets the interested reader should refer to U.S. Provisional Application No. 60/210,908, which is herein expressly incorporated in its entirety by reference.

Since no information regarding what nodes belong to the same scatternet is included in the INQUIRY RESPONSE message, i.e., in an FHS packet, the inquiring node must first establish a connection to the responding node before it can, potentially, find out what other nodes that are reachable through it. Further, even then there is no simple mechanism to collect this information. In addition, establishing connections in order to gather information is a time consuming and resource demanding procedure, which could be greatly simplified if more information could be transferred in the INQUIRY RESPONSE message.

SUMMARY

These and other problems, drawbacks and limitations of conventional techniques are overcome according to the present invention by assigning network identities to each network. During paging procedures nodes can determine whether the responding nodes are members of the same network by comparing network identities.

The present invention provides various apparatus and methods for initial selection of the network identity. In addition, when two networks merge, the present invention provides methods and apparatus for selecting which network identity should be used by the new merged network. Further, to prevent two networks which were once part of the same network from having the same network identity, the present invention provides for periodically changing the network identity. In addition, the present invention allows nodes which have broken a connection between them to determine whether they are still members of the same network before broadcasting a message to change the network identity.

By providing a network identity, nodes in ad-hoc networks can determine how neighbor nodes are interconnected. Further, a node can determine whether a neighbor node is already connected to the same network as the node.

BRIEF DESCRIPTION OF THE DRAWINGS

The objects and advantages of the invention will be understood by reading the following detailed description in conjunction with the drawings in which:

FIG. 1 illustrates an exemplary piconet;

FIG. 2 illustrates an exemplary star-topology network;

FIG. 3 illustrates an exemplary scatternet formed by a plurality of piconets;

FIG. 4 illustrates a conventional Bluetooth packet;

FIG. 5 illustrates signalling between two nodes for neighbor discovery and connection establishment;

FIG. 6 illustrates a conventional FHS packet;

FIGS. 7A-7C illustrate exemplary methods for selecting a scatternet ID in accordance with the present invention;

FIGS. 8A and 8B illustrate exemplary methods for selecting a scatternet ID in accordance with another embodiment of the present invention;

FIG. 9 illustrates an exemplary method for selecting a scatternet ID when a node receives a change-scatternet-ID-message in accordance with one embodiment of the present invention;

FIG. 10 illustrates another exemplary method for selecting a scatternet ID when a node receives a change-scatternet-ID-message in accordance with another embodiment of the present invention;

FIGS. 11A and 11B illustrate exemplary methods for selecting a scatternet ID in accordance with yet another exemplary embodiment of the present invention.

FIG. 11C illustrates an exemplary method for reception and processing of a change-scatternet-ID-message in accordance with exemplary embodiments of the present invention;

FIG. 12 illustrates an exemplary method for periodically selecting a scatternet ID in accordance with one embodiment of the present invention;

FIG. 13 illustrates an exemplary scatternet in accordance with the present invention;

FIG. 14 illustrates an exemplary method for selecting a scatternet ID when a node has broken a connection with a node of another piconet in accordance with another exemplary embodiment of the present invention; and

FIG. 15 illustrates an exemplary Bluetooth unit in accordance with the present invention.

DETAILED DESCRIPTION

The present invention is directed to ad-hoc networks. More particularly, the present invention is directed to obtaining topology information related to an ad-hoc network.

In dynamic networks such as ad-hoc networks keeping track of the detailed topology of a scatternet, i.e., the exact connections that each node in the scatternet has with other nodes in the scatternet, would be too demanding. Further, as a scatternet grows, tracking of the detailed topology of a scatternet becomes increasingly complex. In accordance with exemplary embodiments of the present invention, nodes only keep track of which scatternet a particular node belongs to, so that it can easily be determined whether two nodes belong to the same scatternet, without knowing anything about the connection structure between them. This reduces the topology of a scatternet to the "cloud" stage, i.e., a cloud being a commonly used symbol to represent a network with arbitrary topology.

In accordance with exemplary embodiments of the present invention each scatternet has a randomly chosen scatternet identity (scatternet ID) which is known by each node in the scatternet. When responding to an INQUIRY message, a node will include the scatternet ID in the INQUIRY RESPONSE message, e.g., in the class of device field. The inquiring node then knows that the node from which it has received an INQUIRY RESPONSE message is reachable via all other nodes that respond with the same scatternet ID. By collecting INQUIRY RESPONSE messages including different scatternet IDs, the inquiring node obtains a picture of the different "scatternet islands" that are present in the vicinity of the particular node. In accordance with exemplary embodiments of the present invention, an idle node, i.e., a node that is not connected to any other node, should respond with a default scatternet ID, e.g., all zeros, reserved for this purpose. Alternatively, the idle status could be indicated explicitly by some other means, e.g., using the undefined field of the FHS packet. In such case, an idle node would not have to include any scatternet ID at all in the INQUIRY RESPONSE message.

In the following description references are made to neighbor nodes of a particular node. One skilled in the art will recognize that there are two types of neighbor nodes, those which are connected to the particular node (i.e., part of the same piconet or scatternet) and those which are within radio range of the particular node but does not have an established connection with the particular node. Accordingly, it will be recognized that in the following when it is described that a message is forwarded to a neighbor node, this neighbor node is a connected neighbor node. Further, it will be recognized that when it is described that the particular node is performing a process to determine which network a neighbor node belongs to, this neighbor node is within radio range but does not have an established connection with the particular node. This will be evident because if the particular node and the neighbor node have an established connection, the particular node will already know which network the neighbor node belongs to.

FIG. 7A illustrates an exemplary method for selecting a scatternet ID in accordance with the present invention. When a piconet is first established between a master node and a slave node, the piconet does not have a scatternet ID. Accordingly, the master of the piconet chooses a scatternet ID at random (Step 710). The master then transfers the scatternet ID to the slave (Step 720).

If a new node joins the piconet, this node may be a single node (i.e., idle) or the new node can be connected to another scatternet or piconet, where a single piconet is a trivial case of the more general scatternet concept. In the former case, the scatternet ID should be transferred to the new node. In the latter case two scatternets have been merged and one of their respective scatternet IDs has to be selected as the scatternet ID for the new merged scatternet. Accordingly, it is determined whether a new node has joined the scatternet (Step 730). If a new node has not joined the scatternet ("NO" path out of decision Step 730), then the nodes of the piconet continue to wait (Step 740). If, however, a new node joins the scatternet ("YES" path out of decision Step 730), then it is determined whether the new node was previously idle (Step 745). An idle node is not a member of a scatternet, and hence, does not have a scatternet ID. If the new node was previously idle ("YES" path out of decision Step 745) then the scatternet ID is transferred to the new node (Step 747). If the new node is not an idle node then the new node is a member of another scatternet and the scatternets are merged (Step 750).

If the size of the scatternets are available ("YES" path out of decision Step 760), then the scatternet ID of the larger scatternet is selected as the scatternet ID of the merged scatternets (Step 770). A change-scatternet-ID-message is then broadcast through the scatternet whose scatternet ID was not selected, i.e., the change-scatternet-ID-message is forwarded between nodes until the message has reached all the nodes in the scatternet/network

(Step 790). If, however, the size of the scatternets is not available ("NO" path out of decision Step 760) then the scatternet ID of either of the two nodes forming the connection that merges the two scatternets could be chosen. An arbitrary rule could be selected for this choice, e.g., that the scatternet associated with the joining node is selected (Step 780), wherein the "joining node" is the node assuming the slave role in the connection merging the scatternets. When a scatternet ID has been selected a change-scatternet-ID-message is then broadcast through the scatternet whose scatternet ID was not selected (Step 790).

The method illustrated in FIG. 7A is an advantageous way of selecting a scatternet ID. Since the scatternet ID of the scatternet with the larger number of nodes is selected the change-scatternet-ID-message only has to be broadcast through the smaller scatternet, thus resulting in less load being placed on the ad-hoc network. However, such size information may not be generally available. Further, when a change-scatternet-ID-message is broadcast due to a merger of two scatternets, there is a slight risk that a merger with another scatternet, or several scatternets, is being performed simultaneously. This could result in colliding of change-scatternet-ID-messages, i.e., multiple change-scatternet-ID-messages being distributed through the scatternet simultaneously. Further, a scatternet could be merging with another scatternet via two different connections, i.e., between the same two scatternets, simultaneously. This could result in either colliding or looping of change-scatternet-ID-messages. Accordingly, it would be desirable to provide a mechanism to control these situations, i.e., a mechanism of how to establish which scatternet ID has precedence over the other scatternet ID. This mechanism would govern the way a node acts whenever it has to choose between scatternet IDs, e.g., when it receives a change-scatternet-ID-message and already has a stored scatternet ID. The mechanism would instruct the node whether it should replace the stored scatternet ID with one received in the change-scatternet-ID-message or retain its stored scatternet ID and discard the new one it received.

One way to address this situation is to provide a mechanism that a node always chooses the highest scatternet ID, between the scatternet ID received in the change-scatternet-ID-message and the stored scatternet ID. However, this mechanism has the drawback that with every scatternet ID change the scatternet ID will be driven towards the highest value in the scatternet ID range. This will result in an accumulation of scatternet IDs in the upper part of the scatternet ID range, thereby reducing the randomness and increasing the risk that two scatternets choose the same scatternet ID. To avoid such accumulation, a rule must not tend to drive the scatternet ID towards any particular value in the scatternet ID range. Further, the chosen value should still be completely arbitrary and every part of the scatternet ID range should be equally probable.

FIGS. 7B and 7C illustrate alternative methods for selecting a scatternet ID. Whereas in FIG 7A the scatternet ID can be selected based upon the size of the scatternet or based upon which node is the joining node, the method can be implemented such that the scatternet ID is selected solely based upon which node is the joining node (FIG. 7B), or solely based upon the size of the scatternet (FIG. 7C). It will be recognized that if the selection of the scatternet ID is based solely upon the size of the scatternets, the size of the scatternets will always be available, and hence, the method illustrated in FIG. 7C does not need to determine whether scatternet size is available (Step 760 in FIG. 7A).

FIG. 8A illustrates an exemplary method for selecting a scatternet ID when two scatternets have been merged, without reducing the randomness of the selected scatternet ID and without driving the scatternet ID towards any particular value in the scatternet ID range. It is first determined whether a new node has joined the scatternet (Step 805). If a new node has not joined the scatternet ("NO" path out of decision Step 805) then the method loops back to decision Step 805 and waits for a new node to join the scatternet. If a new node has joined the scatternet ("YES" path out of decision Step 805) then it is determined whether the new node was previously idle (Step 806). If the new node was previously idle ("YES" path out of decision Step 806) then the scatternet ID is transferred to the new node (Step 807). If the new node was not previously idle ("NO" path out of decision Step 806) then the scatternets are merged (Step 810). The nodes then exchange their scatternet IDs (Step 815). Each node then determines whether the received scatternet ID is the same as the stored scatternet ID (Step 820). If the received scatternet ID is the same as the stored scatternet ID ("YES" path out of decision Step 820) then the processing in the node ends (Step 825).

If the scatternet ID is not the same as the stored scatternet ID ("NO" path out of decision Step 820) then an "Exclusive OR" (XOR) operation is performed using the least significant bit (LSB) of the received scatternet ID and the current scatternet ID (Step 830). If the result of the XOR operation is 0 ("YES" path out of decision Step 835) then the node will determine that the lower scatternet ID has precedence over the higher scatternet ID (Step 840). If, however, the result of the XOR operation is not equal to 0 ("NO" path out of decision Step 835) then the node will determine that the higher scatternet ID has precedence over the lower scatternet ID (Step 845).

After it has been determined which scatternet ID has precedence over the other (Step 840 or Step 845) it is determined whether the received scatternet ID has precedence over the old scatternet ID (Step 850). If the received scatternet ID has precedence over the old scatternet ID ("NO" path out of decision Step 850) then the processing for the node ends (Step 855). If, however, the received scatternet ID does not have precedence over the old scatternet ID ("YES" path out of decision Step 850) then the node sends the new scatternet ID in a change-scatternet-ID message to the new node (Step 860). In FIG. 8A the dashed lines returning to step 805 illustrate that the method illustrated in FIG. 8A is performed by nodes as a continuous process.

It will be recognized in the method described above in connection with FIG. 8A that the exchanging of scatternet IDs by the nodes (Step 815) and the sending of a change-scatternet-ID-message to the new node which is to have its scatternet IDs replaced (Step 860) is separated by a period of time. If one or both of these scatternet IDs are changed in between these two operations problems may result. Further, when a node participating in more than one piconet switches between these piconets the length of time between the above-described operations can be significant. To help illustrate this problem, a brief example of a situation in which this problem can occur is presented.

Assume that node A in scatternet A connects to node B in scatternet B. The two nodes then exchange scatternet IDs (Step 815) by using means other than the above described change-scatternet-ID-message, for example, by using some type of message dedicated for this purpose. Further assume, that the scatternet ID of node B has precedence over the scatternet ID of node A as a result of the XOR operation (Step 830 through Step 845). Accordingly, node A will wait for a change-scatternet-ID-message from node B. Assume that while node B is preparing to send a change-scatternet-ID-message to node A, node B receives a change-scatternet-ID-message from its old scatternet, i.e., scatternet B, that takes precedence over the current scatternet ID of node B. If this new scatternet ID does not take precedence over the scatternet ID of node A, then when node B forwards the change-scatternet-ID-message to node A, node A will discard the change-scatternet-ID-message and keep its old scatternet ID. This results in the two old scatternets, although now merged into one, still having different scatternet IDs.

FIG. 8B illustrates an exemplary method for selecting a scatternet ID when two scatternets have merged which addresses the above described problems. Accordingly, whereas in FIG. 8A the nodes exchange scatternet IDs using means other than a change-scatternet-ID-message (Step 815), in FIG. 8B this step is replaced by a step where a node sends a change-scatternet-ID-message to the new node and receives a change-scatternet-ID-message from the new node (Step 817). When these change-scatternet-ID-messages are received they are processed in the same way as any received change-scatternet-ID-message, which effectively means that the exchanging of scatternet IDs and the replacement of one of the scatternet IDs are achieved with the same message, i.e., without time separation. Further, when the received scatternet ID is the same as the stored scatternet ID, the change-scatternet-ID-message is discarded (Step 826).

Another difference between the procedures illustrated FIG 8A and FIG 8B is the procedure following the determination of the result of the XOR operation on the least significant bit of the two scatternet IDs, i.e., decision Step 835. In FIG 8A the node just determines which of the two scatternet IDs that has precedence over the other (Step 840 or 845) without replacing the old scatternet ID irrespective of the result of this determination. If the stored scatternet ID has precedence over the received scatternet ID, the node sends a change-scatternet-ID-message including the stored scatternet ID to the other node (Step 860). The actual replacement of scatternet ID will occur when this change-scatternet-ID-message is received unless the above described problem occurs.

In contrast, in the procedure illustrated in FIG 8B, based on the result of the XOR operation (decision Step 835) the node selects one of the scatternet IDs to be the new one (Steps 841 or 846) and if the received scatternet ID is selected, the old scatternet ID is replaced by the received scatternet ID and the change-scatternet-ID-message is forwarded to other nodes in the original scatternet of the node (Step 851 and 861). On the other hand, if the received scatternet ID is not selected as the new scatternet ID, the node keeps the old scatternet ID and discards the change-scatternet-ID-message. By sending a change-scatternet-ID-message to the new node and receiving a change-scatternet-ID-message from the new node (Step 817) the problems described above in connection with FIG. 8A are avoided because the exchanging of scatternet IDs and the replacement of scatternet ID are effectuated by the same message, i.e. without separation of the two steps. It will be recognized that in FIG. 8B the dashed lines returning to step 805 illustrate that the method illustrated in FIG. 8B is performed by nodes as a continuous process.

FIG. 9 illustrates the exemplary behavior of a node which has received a change-scatternet-ID-message. The node determines whether it has received a change-scatternet-ID-message (Step 905). If a node has not received a change-scatternet-ID-message ("NO" path out of decision Step 905) the node continues to wait to receive a change-scatternet-ID-message. If, however, the node has received a change-scatternet-ID-message ("YES" path out of decision Step 905) the node determines whether the received scatternet ID is the same as the stored scatternet ID (Step 910). If the received scatternet ID is the same as the stored scatternet ID ("YES" path out of decision Step 910) the node will discard the change-scatternet-ID-message (Step 915). If the received scatternet ID is not the same as the stored scatternet ID ("NO" path out of decision Step 910) then the node performs an XOR operation using the least significant bit from the scatternet ID in the message and the current stored scatternet ID (Step 920).

If the result of the XOR operation is equal to 0 ("YES" path out of decision Step 925) the node will select the lower scatternet ID as the new scatternet ID (Step 930). If the result of the XOR operation is not equal to 0 ("NO" path out of decision Step 925) then the node will select the higher scatternet ID as the new scatternet ID (Step 935). Once either the lower scatternet ID or the higher scatternet ID has been selected (in accordance with either Step 930 or Step 935) then the node determines whether the received scatternet ID has replaced the old scatternet ID (Step 940). If the received scatternet ID has not replaced the old scatternet ID ("NO" path out of decision Step 940) then the node keeps the old scatternet ID and discards the change-scatternet-ID-message (Step 945). If, however, the received scatternet ID has replaced the old scatternet ID ("YES" path out of decision Step 940) then the node stores the new scatternet ID and forwards the change-scatternet-ID-message to all connected neighbor nodes except the one from which the change-scatternet-ID-message was received (Step 950).

FIG. 10 illustrates another exemplary behavior of a node which has received a change-scatternet-ID-message. In accordance with this embodiment of the present invention, a timestamp, representing the time when the scatternet ID was generated, is associated with each scatternet ID. This timestamp is included in the change-scatternet-ID-message together with the new scatternet ID. A node which receives a change-scatternet-ID-message compares the received timestamp with a timestamp associated with the stored scatternet ID.

Initially, a node determines whether it has received a change-scatternet-ID-message

(Step 1005). If the node has not received a change-scatternet-ID-message ("NO" path out of decision Step 1005) then the node waits until it has received a change-scatternet-ID-message. If a node has received a change-scatternet-ID-message ("YES" path out of decision Step 1005) then the node determines whether the scatternet ID in the received message is the same as the stored scatternet ID (Step 1010). If the received scatternet ID is the same as the stored scatternet ID ("YES" path out of decision Step 1010) then the received message is discarded (Step 1015).

If the received scatternet ID is not the same as the stored scatternet ID ("NO" path out of decision Step 1010) then the node compares the received timestamp with the stored timestamp (Step 1025). If the timestamps are the same ("YES" path out of decision Step 1030) then the node performs an XOR operation using the least significant bit of the scatternet ID in the message and the current scatternet ID (Step 1050). If the timestamps are not the same ("NO" path out of decision Step 1030) then the node determines whether the received timestamp is newer than the stored timestamp (Step 1035). If the received timestamp is not newer than the stored timestamp ("NO" path out of decision Step 1035), i.e., indicating that the stored timestamp is "fresher" than the received timestamp, then the node will discard the message (Step 1015). If, however, the received timestamp is newer than the stored timestamp ("YES" path out of decision Step 1035) then the node replaces the stored scatternet ID with the received scatternet ID (Step 1040). The node will then forward the change-scatternet-ID-message to all connected neighbor nodes except the one from which the change-scatternet-ID-message was received (Step 1045).

If the timestamps are the same ("YES" path out of Step 1030) and the node has performed an XOR operation using the least significant bit of the scatternet ID in the message and the current scatternet ID (Step 1050) then the node will determine the result of the XOR operation (Step 1055). If the result of the XOR operation is equal to 0 ("YES" path out of decision Step 1055) the node selects the lower scatternet ID as the new scatternet ID (Step 1060). If the result of the XOR operation is not equal to 0 ("NO" path out of decision Step 1055) the node will select the higher scatternet ID as the new scatternet ID (Step 1065).

Once the node has selected the higher scatternet ID (Step 1065) or the lower scatternet ID (Step 1060), the node determines whether the received scatternet ID has replaced the old scatternet ID (Step 1070). If the received scatternet ID has not replaced the old scatternet ID ("NO" path out of decision Step 1070) then the node keeps the old scatternet ID and discards the change-scatternet-ID-message (Step 1075). If, however, the received scatternet ID has replaced the old scatternet ID ("YES" path out of decision Step 1070) then the node stores the new scatternet ID and forwards the change-scatternet-ID-message to all connected neighbor nodes except the one from which the change-scatternet-ID-message was received (Step 1080).

The timestamp method illustrated in FIG. 10 helps avoid colliding or looping of change-scatternet-ID-messages because when a node receives a change-scatternet-ID-message with a timestamp which is older than the stored timestamp associated with the node's current scatternet ID (Step 1035) the node discards the received change-scatternet-ID-message. Accordingly, older scatternet IDs contained in change-scatternet-ID-messages are prevented from propagating throughout the remainder of the scatternet.

A corollary to the selection of a single scatternet ID when two separate scatternets merge, is that when a single scatternet is split, the two resulting scatternets should each have a different scatternet ID. FIGS. 11A and 11B illustrate exemplary methods of periodically generating and propagating new scatternet IDs throughout a scatternet, thereby ensuring if a scatternet has broken off of the current scatternet that the original scatternet and the broken off scatternet will have different scatterent IDs. Preferably, only the master nodes periodically generate and distribute scatternet IDs. This spares the slave nodes from this burden and reduces the risk of colliding messages distributing periodically generated scatternet IDs. However, it would be quite possible to let all the nodes periodically generate and distribute new scatternet IDs. In addition to handling the split scatternet case, the periodical generation and distribution of new scatternet IDs also work as a general "safety net" procedure which cleans up fault conditions that may occur during the management of scatternet IDs.

According to the method illustrated in FIGS. 11A and 11B, master nodes, but not slave nodes, will periodically generate and broadcast new random scatternet IDs. Accordingly, each node determines whether it is a master node (Step 1105). If a node is not a master node ("NO" path out of decision Step 1105) then the processing for that node ends (Step 1110). If a node is a master node ("YES" path out of decision Step 1105) then the node randomly chooses a time value between a predefined minimum value T0 and a predefined maximum value T1 (Step 1115). The node then starts a timer (Step 1120).

If the timer has not exceeded the random time value ("NO" path out of decision Step 1125) then the node determines whether a new scatternet ID has been received and whether the stored scatternet ID has changed since the timer was started (Step 1130). If a new scatternet ID has been received and the stored scatternet ID has been changed since the timer was started ("YES" path out of decision Step 1130) then the node randomly chooses a time value (Step 1115). If, however, a new scatternet ID has not been received and the stored scatternet ID has not been changed since the timer was started ("NO" path out of decision Step 1130) the node continues to determine whether the timer has exceeded the random value (Step 1125). If the timer exceeds the random time value ("YES" path out of decision Step 1125) then the node generates a new random scatternet ID (Step 1135).

The differences between the procedures illustrated in FIGS. 11A and 11B is the type of message distributed by the master node after the master node has generated a new random ID (Step 1135). In accordance with the method illustrated in FIG. 11A after the master node has generated a new random ID (Step 1135) the master node distributes the new ID in a change-scatternet-ID-message (Step 1140) and then returns to randomly choose a time value (Step 1115). A node which receives the new ID in the change-scatternet-ID-message generated using the method described in connection with FIG. 11A would perform an XOR operation on the least significant bit of the scatternet ID in the message and the current ID as described in above in connection with FIGS. 8A or 8B, i.e., Steps 830-860 of FIG. 8A or Steps 830-862 in FIG. 8B. Alternatively, if the timestamp method described in connection with FIG. 10 is used, a node receiving the change-scatternet-ID-message generated using the method described in connection with FIG. 11A would follow steps 1010-1080 in FIG. 10.

In the method illustrated in FIG. 11B after generating a new random ID (Step 1136) the master node distributes the new ID in a periodically-changed-scatternet-ID-message (Step 1142) and then returns to randomly choose a time value (Step 1115). The differences between a change-scatternet-ID-message and a periodically-changed-scatternet-ID-message is the procedures followed by a node which receives the particular message. It should be noted that the procedure described in connection with 11B can be used only if the timestamp method is used, i.e., if the procedure described in connection with FIG. 10 is used for reception of regular change-scatternet-ID-messages.

FIG. 11C illustrates the procedures performed by a node which receives a periodically-changed-scatternet-ID-message. Initially, the node determines whether it has received a periodically-changed-scatternet-ID-message (Step 1145). If the node has not received a periodically-changed-scatternet-ID-message ("NO" path out of decision Step 1145) then the node continues to wait until it receives such a message. If the node does receive a periodically-changed-scatternet-ID-message ("YES" path out of decision Step 1145) the node compares the timestamp in the received message with the stored timestamp (Step 1150) and the node determines whether the timestamps are the same (Step 1155). If the node determines that the timestamps are the same ("YES" path out of decision Step 1155) the node uses the XOR method to select a scatternet ID (Step 1160) as described above in connection with FIGS. 8-10, i.e., Steps 830-861 of FIG. 8B; Steps 920-950 of FIG. 9; and Steps 1050-1080 of FIG. 10 with the only difference being that the message to be either discarded or forwarded is a periodically-change-scatternet-ID-message instead of a regular change-scatternet-ID-message.

If it is determined that the timestamps are not the same ("NO" path out of decision Step 1155) then the node determines whether the differences between the timestamps is less than a time period T0 (Step 1162), wherein T0 is the shortest time that the timer governing periodical generation of new scatternet IDs can be set to. If the differences between the timestamps is less than T0 ("YES" path out of decision step 1162) then it is determined whether the received timestamp is older than the stored timestamp (Step 1164). If the received timestamp is not older than the stored timestamp ("NO" path out of decision Step 1164) then the node discards the message (Step 1166). If the received timestamp is older than the stored timestamp ("YES" path out of decision Step 1164) then the node replaces the stored scatternet ID with the received scatternet ID (Step 1175) and forwards the periodically-changed-scatternet-ID-message (Step 1180).

If the differences between the timestamps is not less than the time period T0 ("NO" path out of decision Step 1162) then the node determines whether the received timestamp is newer than the stored timestamp (Step 1168). If the received timestamp is not newer than the stored timestamp ("NO" path out of decision Step 1168) then the node discards the message (Step 1170). If, however, the received timestamp is newer than the stored timestamp ("YES" path out of decision Step 1168) then the node replaces the stored scatternet ID with the received scatternet ID (Step 1175) and forwards the periodically-changed-scatternet-ID-message (Step 1180).

It will be recognized that using the periodically-changed-scatternet-ID-message described in connection with FIGS. 11B and 11C results in the scatternet ID in the periodically-changed-scatternet-ID-message replacing the stored scatternet ID. Further, in cases of colliding periodically-changed-scatternet-ID-messages, the message with the oldest of the two timestamps will prevail because this scatternet ID will likely have spread over a larger part of the scatternet.

FIG. 12 illustrates an exemplary method for generating a new random ID when only the XOR method (without timestamps) is used. The procedure described in connection with FIG. 11A governs when to generate a new random scatternet ID and the procedure described in connection with FIG. 9 is used when a node receives a change-scatternet-ID-message. The method illustrated in FIG. 12 provides the details of Step 1135 in FIG. 11A. In accordance with the method illustrated in FIG. 12 a fraction F is chosen such that $1/2^n \leq F \leq ½$ where n is the number of bits in the scatternet ID. As will be described in more detail below the fraction F is used in the generation of new random scatternet IDs to prevent these random scatternet IDs from being chosen from a too small part of the total scatternet ID range, which would degrade the random properties of the scatternet ID. Accordingly, a node first determines whether the stored scatternet ID is in the lowest F fraction of the range of scatternet ID (Step 1243). If the stored scatternet ID is in the lowest F fraction of the range of scatternet ID ("YES" path out of decision Step 1243) the node will set the least significant bit of the new scatternet ID to the inverted value of the least significant bit of the stored scatternet ID (Step 1246). Next the node will randomly select the remaining bits of the new scatternet ID in the range above the stored scatternet ID (Step 1249).

If the stored scatternet ID is not in the lowest F fraction of its range ("NO" path out of decision Step 1243) then the node determines whether the stored scatternet ID is in the highest F fraction of the range of scatternet IDs (Step 1252). If the stored scatternet ID is in the highest F fraction of the range of scatternet IDs ("YES" path out of decision Step 1252) then the node sets the least significant bit of the new scatternet ID to the same value as the least significant bit of the stored scatternet ID (Step 1255). Next the node randomly selects the remaining bits of the new scatternet ID in the range below the stored scatternet ID (Step 1258).

If the stored scatternet ID is not in the highest F fraction of the range of scatternet

IDs ("NO" path out of decision Step 1252) then the node randomly chooses the least significant bit of the new random ID to be either a 1 of a 0 (Step 1261). Next the node performs an XOR operation using the randomly chosen least significant bit with the least significant bit of the stored scatternet ID (Step 1264). If the result of the XOR operation is 0 ("NO" path out of decision Step 1267) then the node randomly selects the remaining bits of the new scatternet ID in the range below the stored scatternet ID (Step 1258). If, however, the result of the XOR operation is 1 ("YES" path out of decision Step 1267) then the node randomly selects the remaining bits of the new scatternet ID in the range above the stored scatternet ID (Step 1249). It will be recognized that when F=1/2 that the scatternet ID will satisfy either step 1243 or step 1252, and hence, steps 1261-1267 would not be needed.

Another way of dealing with the problem of keeping the scatternet IDs unique in case of a split scatternet is to try to detect when a scatternet is split into two separate scatternets and then generate a new scatternet ID in one of the separate scatternets. A scatternet split is always caused by the breaking of a connection between two nodes. If the broken connection was the only connection between two parts of the scatternet, the broken connection results in two separate scatternets. However, a broken connection does not always split a scatternet. Other connections may still hold all the nodes together and in such case they are still a single scatternet. Since a broken connection between two nodes of a scatternet does not necessarily indicate the formation of two separate scatternets, it would be advantageous to be able to determine whether in fact two scatternets have been formed as a result of the split between the two nodes. For example, referring now to FIG. 13, if connection 1305 between nodes 1310 and 1320 were broken, the two nodes would not be able to communicate with each other directly over link 1305. However, nodes 1310 and 1320 are still part of the same scatternet, i.e., they are still connected by nodes 1330

1405) the node continues to make this determination. If the node determines that it has broken a connection with another node ("YES" path out of decision Step 1405) the node determines whether the node is now idle (Step 1407). If the node is now idle ("YES" path out of decision step 1407) then processing for the node ends (Step 1408). If the node is not idle ("NO" path out of decision step 1407) then the node determines whether it has a lower BD_ADDR than the node with which there was a broken connection (Step 1410). If the node does not have a lower BD_ADDR than the node with which there was a broken connection ("NO" path out of decision Step 1410) the node sets a timer (Step 1412) and waits for a message from the other node of the broken connection (Step 1415). The node with the higher BD_ADDR then determines whether it has received a message from the node with the lower BD_ADDR (Step 1420). If the node with the higher BD_ADDR has not received a message from the node with the lower BD_ADDR ("NO" path out of decision Step 1420) then the node determines whether the timer has expired (Step 1422). If the node determines that the timer has expired ("YES" path out of decision step 1422) then the node concludes that the scatternet has been split (Step 1424). If the timer has not expired ("NO" path out of decision step 1422) then the node continues to wait to receive the message (Step 1415). If, however, the node with the higher BD_ADDR has received a message from the node with the lower BD_ADDR ("YES" path out of decision Step 1420) the node with the higher BD_ADDR sends a reply message to the node with the lower BD_ADDR (Step 1425). Accordingly, these two nodes will now know that although there is no longer a direct link between them, i.e., a link with zero hops, they are still part of the same scatternet.

An alternative to steps 1412-1425 is that the processing in the node with the higher BD_ADDR ends with the "NO" path out of decision step 1410. If a message is subsequently received from the node with the lower BD_ADDR, the node with the higher BD_ADDR would still send a reply message to the node with the lower BD_ADDR, not because the node with the higher BD_ADDR was waiting for the message from the node with the lower BD_ADDR, but because the message from the node with the lower BD_ADDR was a type of message that should always be replied to by the intended receiver, e.g. a route request message. In this alternative the node with the higher BD_ADDR might not determine whether the scatternet was split or not by the broken connection.

If the node determines it has a lower BD_ADDR than the node with which there was a broken connection ("YES" path out of decision Step 1410) the node sends a message to the other node (Step 1430). For example, the message can be a request for route message to the other node. The node then sets a timer (Step 1435) and determines whether it has received a reply message from the other node (Step 1440). If the node has received a reply message from the other node ("YES" path out of decision Step 1440) then the node knows that the other node is part of the same scatternet and the node ends its current processing (Step 1445). If, however, the node does not receive a reply message from the other node ("NO" path out of decision Step 1440) the node determines whether the timer has expired (Step 1450). If the node determines that the timer has not expired ("NO" path out of decision Step 1450) then the node continues to determine whether it has received a reply message from the other node (Step 1440). If the node determines that the timer has expired ("YES" path out of decision Step 1450) then the node determines that the scatternet has been split and generates a new random scatternet ID (Step 1455). The new scatternet ID is then transferred to connected neighboring nodes in a change-scatternet-ID-message (Step 1460).

FIG. 15 illustrates an exemplary apparatus for implementing the methods described above. The apparatus includes antenna 1510, transceiver 1520 and processor 1530. As described above, ad-hoc networking nodes, and Bluetooth units in particular, can be embodied in various forms. Accordingly, the apparatus illustrated in FIG. 15 could be a mobile telephone, a personal digital assistant (PDA), a router, a printer, a kitchen appliance, or any other device for which it would be advantageous to form wireless networks. Further, although FIG. 15 illustrates processor 1530, one skilled in the art will recognize that this apparatus can be implemented with either a microprocessor, an application specific integrated circuit (ASIC), hard-wired logic, or any other equivalent means.

Although the present invention has been described above in one embodiment as one node using a scatternet ID to determine whether another is part of the same scatternet, it will be recognized that there are many other uses for the scatternet ID. For example, one node may broadcast the scatternet ID in a neighbor discovery message and compare the responses received from neighboring nodes. Based upon the comparison of the responses the one node can determine whether the responding nodes are part of the same scatternet.

This information can be used by the one node to determine that if it connects to one of the responding nodes, the one node will also be able to reach the other responding node because they are part of the same scatternet.

The present invention has been described with reference to several exemplary embodiments. However, it will be readily apparent to those skilled in the art that it is possible to embody the invention in specific forms other than those of the exemplary embodiments described above. This may be done without departing from the spirit of the invention. These exemplary embodiments are merely illustrative and should not be considered restrictive in any way. The scope of the invention is given by the appended claims, rather than the preceding description, and all variations and equivalents which fall within the range of the claims are intended to be embraced therein.

WHAT IS CLAIMED IS:

1. In an ad-hoc network including a first node which belongs to a first network, a method for determining a network which a second node belongs to, the method comprising the steps of:

transmitting from the first node a neighbor discovery message;

receiving from the second node a neighbor discovery response message, wherein the neighbor discovery response message includes an identification of a network which the second node belongs to; and

determining which network the second node belongs to using the identification.

2. The method of claim 1, wherein the step of determining further comprises the steps of:

comparing the identification received from the second node with an identification stored in the first node;

determining that the second node belongs to a different network if the received identification does not match the stored identification.

3. The method of claim 1, wherein the ad-hoc network is a wireless network.

4. The method of claim 1, wherein the ad-hoc network operates according to Bluetooth protocol.

5. In an ad-hoc network a method for selecting a network identification comprising the steps of:

selecting by a first node a random network identification;

transferring from the first node to a second node the random network identification;

determining by the second node whether to adopt the random network identification; and

sending a new network identification message if the second node adopts the random network identification.

6. The method of claim 5, wherein the second node adopts the random network identification if a size of a network associated with the random network identification is larger than a size of a network associated with the second node.

7. The method of claim 5, wherein the step of determining comprises the steps of:

performing an exclusive OR operation using a least significant bit of the random network identity and a least significant bit of a network identity associated with the second node; and

selecting the higher network identity if a result of the exclusive OR operation has a value of one.

8. The method of claim 5, wherein the step of determining comprises the steps of:

comparing a timestamp associated with the random network identity and a timestamp associated with a network identity associated with the second node; and

selecting the network identity with the newer timestamp.

9. The method of claim 5, wherein the step of determining comprises the steps of:

determining whether a message which contains the random network identity is a periodic change of identification message;

comparing a timestamp associated with the random network identity and a timestamp associated with a network identity of the second node;

determining whether the timestamp associated with the random network identity is less than a predetermined amount of time different than the timestamp associated with the network identity of the second node;

if the timestamp associated with the random network identity is less than a predetermined amount of time different than the timestamp associated with the network identity of the second node, selecting the network identity with the older timestamp if the message is a periodic change of identification message

if the timestamp associated with the random network identity is greater than a predetermined amount of time different than the timestamp associated with the network identity of the second node, selecting the network identity with the newer timestamp if the message is a periodic change of identification message.

10. The method of claim 5, wherein the step of selecting comprises the steps of:

determining whether the first node has broken a connection with another node;

determining whether the first node has a lower address than the another node;

if the first node has broken a connection with the another node and the first node has a lower address, the first node performs the following steps

sending a message to the another node;

setting a timer;

determining whether the first node has received a response from the another node; and

generating a new random network identification if the first node has not received a response from the another node.

11. In an ad-hoc network a method for selecting a network identification comprising the steps of:

starting a timer;

randomly selecting a time value;

generating a new random network identification if the timer exceeds the randomly selected time value; and

distributing the new network identification to other nodes in the ad-hoc network.

12. The method of claim 11, wherein the new network identification is distributed in a message which includes an indication that the new network identification is a periodically generated network identification.

13. In an ad-hoc network including a first node which belongs to a first network, a method for determining whether a second node belongs to the first network, the method comprising the steps of:

determining in the first node whether a connection with the second node has been broken;

sending a message from the first node to the second node through the ad-hoc network;

setting a timer in the first node; and

determining that the second node does not belong to the first network if the first node does not receive a reply from the second node before the expiration of the timer.

14. The method of claim 13, wherein if it is determined that the second node does not belong to the first network, the first node performs the steps of:

generating a new random network identification; and

distributing the new network identification to other nodes in the first network.

15. An ad-hoc network including a first node which belongs to a first network, the first node comprising:

means for transmitting a neighbor discovery message;

means for receiving from a second node a neighbor discovery response message, wherein the neighbor discovery response message includes an identification of a network which the second node belongs to; and

means for determining which network the second node belongs to using the identification.

16. The first node of claim 15, wherein the means for determining further comprises:

means for comparing the identification received from the second node with an identification stored in the first node;

means for determining that the second node belongs to a different network if the received identification does not match the stored identification.

17. The first node of claim 15, wherein the ad-hoc network is a wireless network.

18. The first node of claim 15, wherein the ad-hoc network operates according to Bluetooth protocol.

19. An ad-hoc network comprising:

means for selecting by a first node a random network identification;

means for transferring from the first node to a second node the random network identification;

means for determining by the second node whether to adopt the random network identification; and

means for sending a new network identification message if the second node adopts the random network identification.

20. The ad-hoc network of claim 19, wherein the second node adopts the random network identification if a size of a network associated with the random network identification is larger than a size of a network associated with the second node.

21. The ad-hoc network of claim 19, wherein the means for determining comprises:

means for performing an exclusive OR operation using a least significant bit of the random network identity and a least significant bit of a network identity associated with the second node; and

means for selecting the higher network identity if a result of the exclusive OR operation has a value of one.

22. The ad-hoc network of claim 19, wherein the means for determining comprises:

means for comparing a timestamp associated with the random network identity and a timestamp associated with a network identity associated with the second node; and means for selecting the network identity with the newer timestamp.

23. The ad-hoc network of claim 19, wherein the means for determining comprises:

means for determining whether a message which contains the random network identity is a periodic change of identification message;

means for comparing a timestamp associated with the random network identity and a timestamp associated with a network identity of the second node; and

means for determining whether the timestamp associated with the random network identity is less than a predetermined amount of time different than the timestamp associated with the network identity of the second node,

wherein if the timestamp associated with the random network identity is less than a predetermined amount of time different than the timestamp associated with the network identity of the second node, the network identity with the older timestamp is selected if the message is a periodic change of identification message,

wherein if the timestamp associated with the random network identity is greater than a predetermined amount of time different than the timestamp associated with the network identity of the second node, the network identity with the newer timestamp is selected if the message is a periodic change of identification message.

24. The ad-hoc network of claim 19, wherein the means for selecting comprises:

means for determining whether the first node has broken a connection with another node;

means for determining whether the first node has a lower address than the another node,

wherein if the first node has broken a connection with the another node and the first node has a lower address, the first node sends a message to the another node, sets a timer, determines whether the first node has received a response from the another node and generates a new random network identification if the first node has not received a response from the another node.

25. In an ad-hoc network a node comprising:

means for starting a timer;

means for randomly selecting a time value;

means for generating a new random network identification if the timer exceeds the randomly selected time value; and

means for distributing the new network identification to other nodes in the ad-hoc network.

26. The node of claim 25, wherein the new network identification is distributed in a message which includes an indication that the new network identification is a periodically generated network identification.

27. In a first ad-hoc network comprising:

means for determining in a first node whether a connection with a second node has been broken;

means for sending a message from the first node to the second node through the first ad-hoc network;

means for setting a timer in the first node; and

means for determining that the second node does not belong to the first ad-hoc network if the first node does not receive a reply from the second node before the expiration of the timer.

28. The ad-hoc network of claim 27, wherein if it is determined that the second node does not belong to the first ad-hoc network, the first node generates a new random network identification and distributes the new network identification to other nodes in the first ad-hoc network.

29. In an ad-hoc network including a first node which belongs to a first network, a method for determining whether a second node and a third node belong to the same network, the method comprising the steps of:

distributing from the first node a neighbor discovery message to the second and third nodes;

receiving from the second and third nodes a neighbor discovery response message, wherein the neighbor discovery response message includes an identification of a network which the second and third nodes belong to; and

determining whether the second node and third node belong to the same network using the identification.

30. The method of claim 29, wherein the step of determining further comprises the steps of:

comparing the identification received from the second node with the identification received from the third node; and

determining that the second node and the third node belong to different networks if the identification received from the second node does not match the identification received from the third node.

31. The method of claim 29, wherein the ad-hoc network is a wireless network.

32. The method of claim 29, wherein the ad-hoc network operates according to Bluetooth protocol.

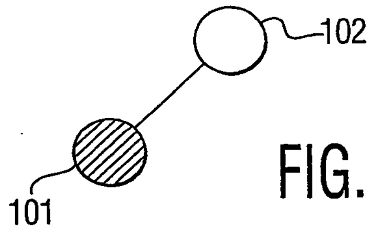


FIG. 1

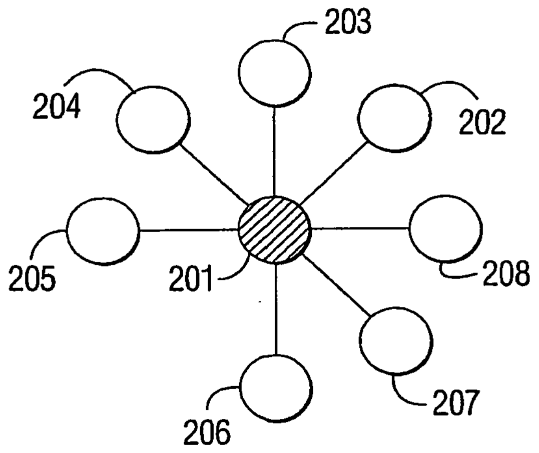


FIG. 2

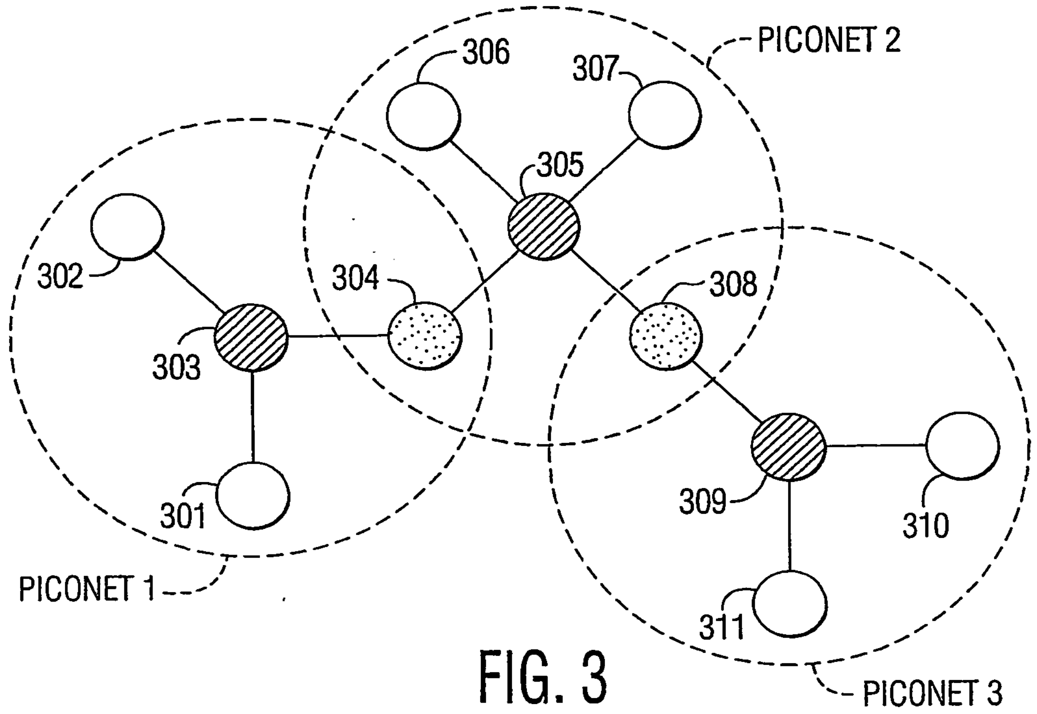


FIG. 3

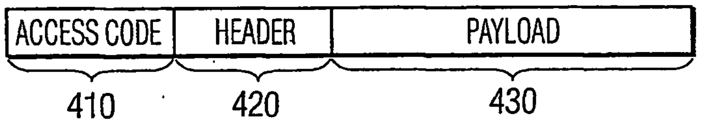


FIG. 4

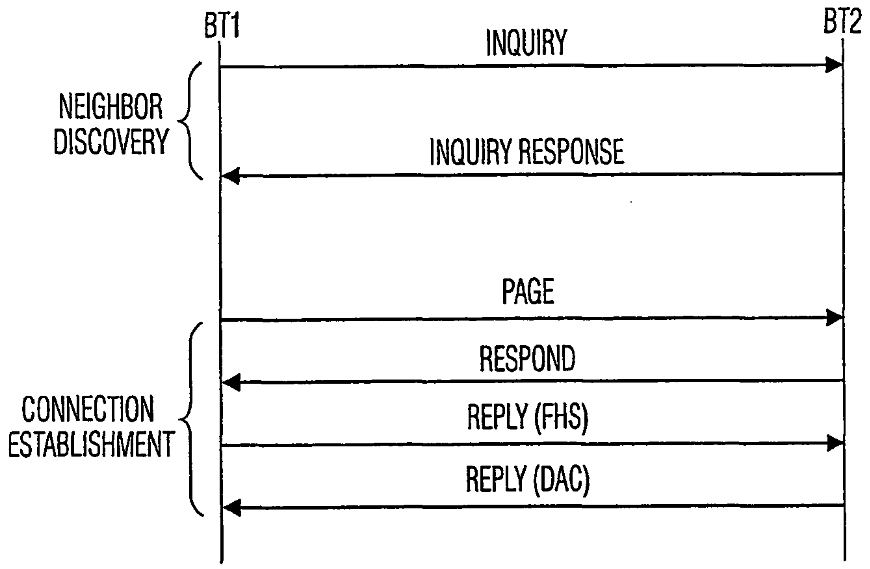


FIG. 5

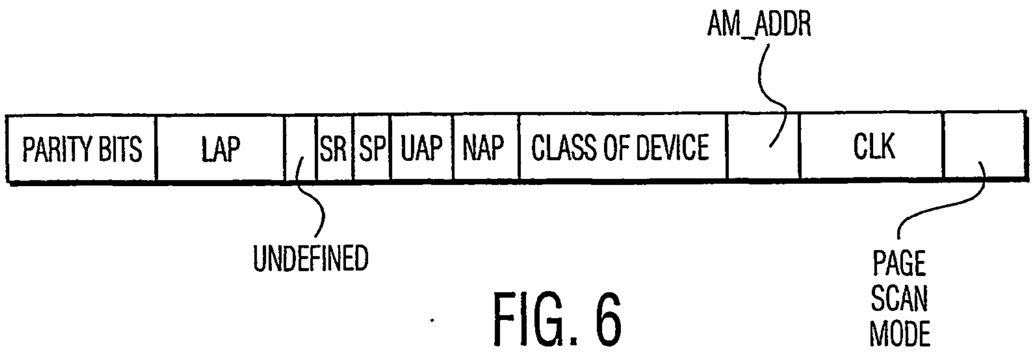


FIG. 6

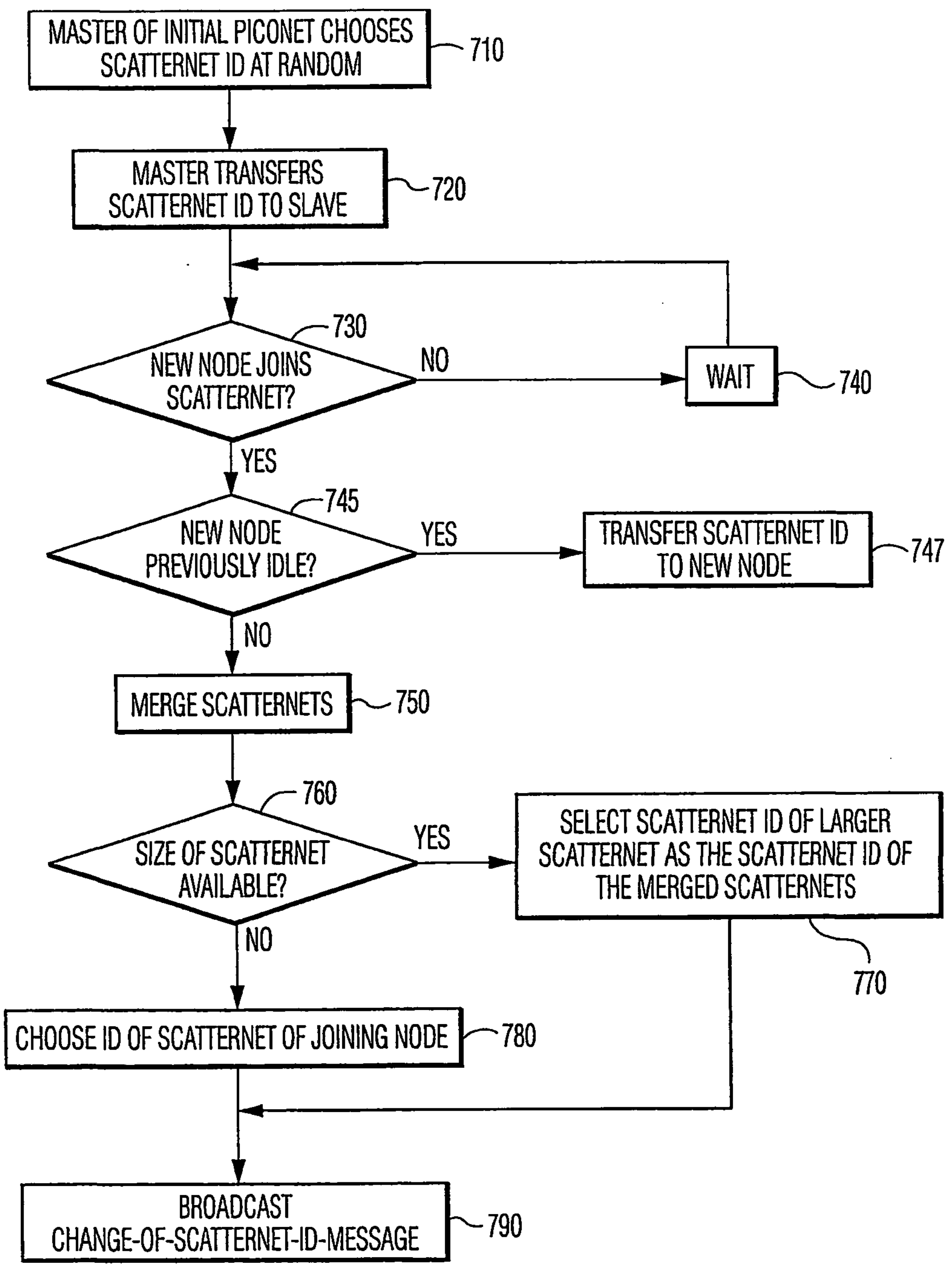

MASTER OF INITIAL PICONET CHOOSES SCATTERNET ID AT RANDOM
710
MASTER TRANSFERS SCATTERNET ID TO SLAVE
720
NEW NODE JOINS SCATTERNET?
730
NO
WAIT
740
YES
NEW NODE PREVIOUSLY IDLE?
745
YES
TRANSFER SCATTERNET ID TO NEW NODE
747
NO
MERGE SCATTERNETS
750
SIZE OF SCATTERNET AVAILABLE?
760
YES
SELECT SCATTERNET ID OF LARGER SCATTERNET AS THE SCATTERNET ID OF THE MERGED SCATTERNETS
770
NO
CHOOSE ID OF SCATTERNET OF JOINING NODE
780
BROADCAST CHANGE-OF-SCATTERNET-ID-MESSAGE
790

FIG. 7A

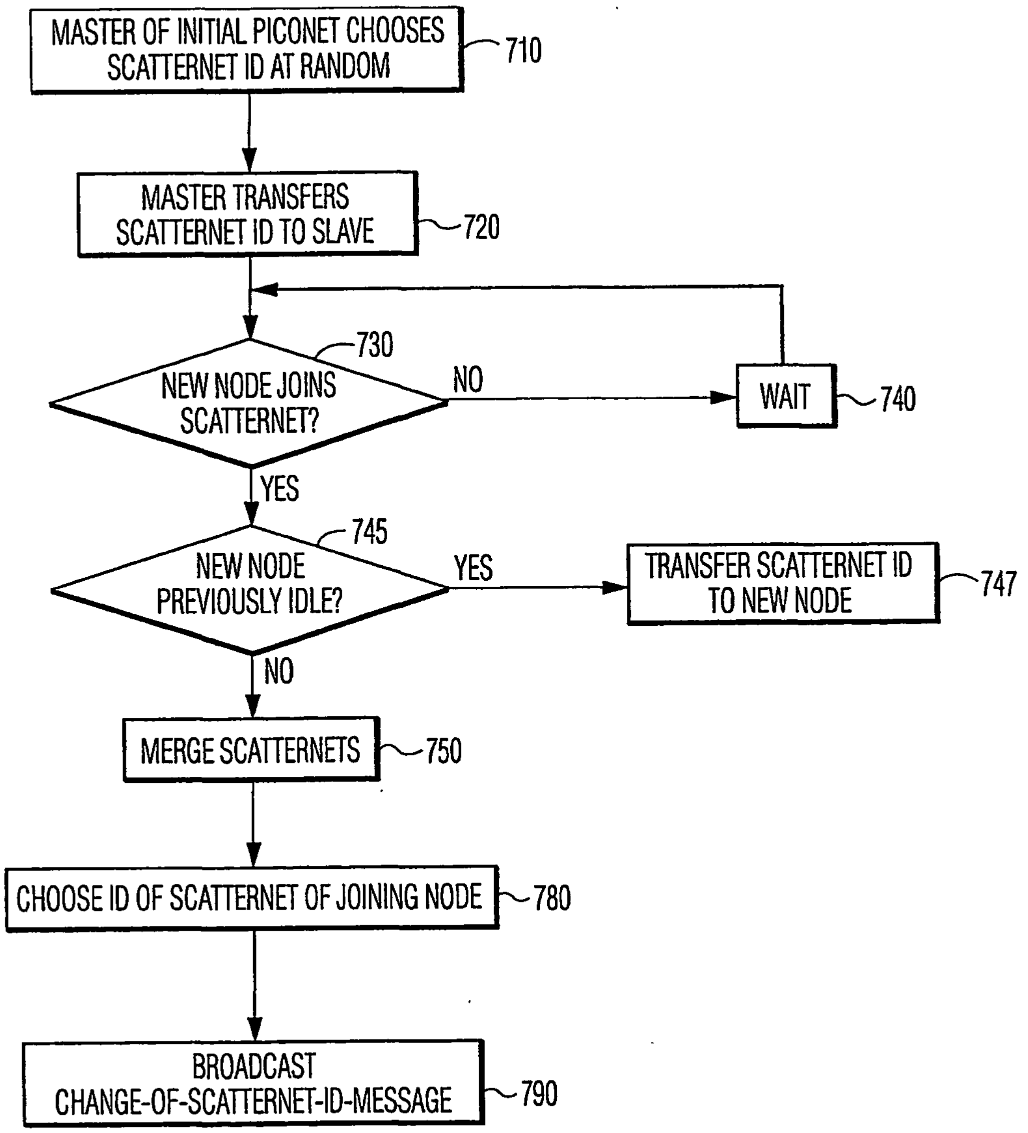

MASTER OF INITIAL PICONET CHOOSES SCATTERNET ID AT RANDOM
710
MASTER TRANSFERS SCATTERNET ID TO SLAVE
720
730
NEW NODE JOINS SCATTERNET?
NO
WAIT
740
YES
745
NEW NODE PREVIOUSLY IDLE?
YES
TRANSFER SCATTERNET ID TO NEW NODE
747
NO
MERGE SCATTERNETS
750
CHOOSE ID OF SCATTERNET OF JOINING NODE
780
BROADCAST CHANGE-OF-SCATTERNET-ID-MESSAGE
790

FIG. 7B

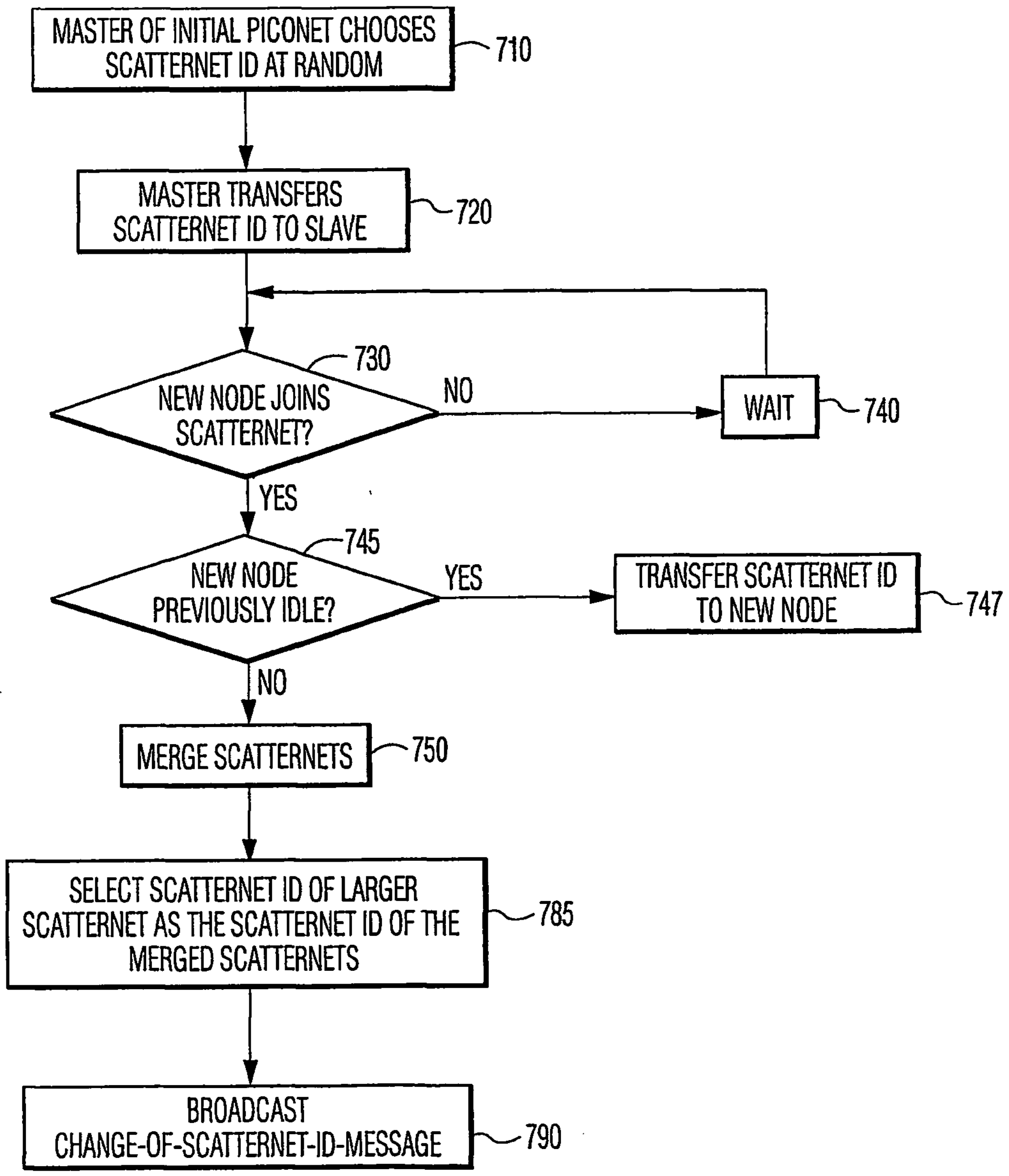
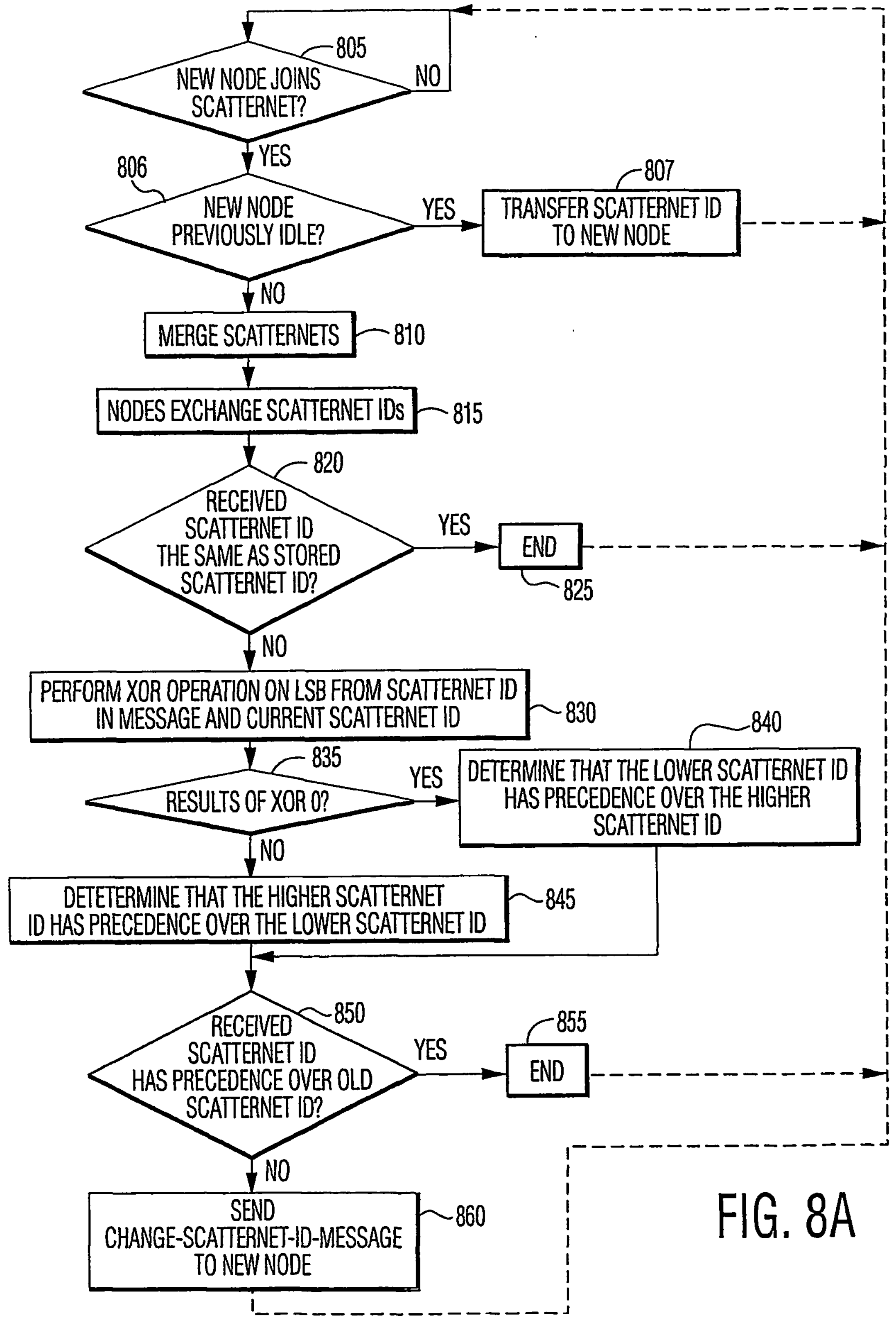

MASTER OF INITIAL PICONET CHOOSES SCATTERNET ID AT RANDOM
710
MASTER TRANSFERS SCATTERNET ID TO SLAVE
720
NEW NODE JOINS SCATTERNET?
730
NO
WAIT
740
YES
NEW NODE PREVIOUSLY IDLE?
745
YES
TRANSFER SCATTERNET ID TO NEW NODE
747
NO
MERGE SCATTERNETS
750
SELECT SCATTERNET ID OF LARGER SCATTERNET AS THE SCATTERNET ID OF THE MERGED SCATTERNETS
785
BROADCAST CHANGE-OF-SCATTERNET-ID-MESSAGE
790

FIG. 7C



805
NEW NODE JOINS SCATTERNET?
NO
YES
806
NEW NODE PREVIOUSLY IDLE?
YES
807
TRANSFER SCATTERNET ID TO NEW NODE
NO
MERGE SCATTERNETS
810
NODES EXCHANGE SCATTERNET IDs
815
820
RECEIVED SCATTERNET ID THE SAME AS STORED SCATTERNET ID?
YES
END
825
NO
PERFORM XOR OPERATION ON LSB FROM SCATTERNET ID IN MESSAGE AND CURRENT SCATTERNET ID
830
835
RESULTS OF XOR 0?
YES
840
DETERMINE THAT THE LOWER SCATTERNET ID HAS PRECEDENCE OVER THE HIGHER SCATTERNET ID
NO
DETETERMINE THAT THE HIGHER SCATTERNET ID HAS PRECEDENCE OVER THE LOWER SCATTERNET ID
845
850
RECEIVED SCATTERNET ID HAS PRECEDENCE OVER OLD SCATTERNET ID?
YES
855
END
NO
SEND CHANGE-SCATTERNET-ID-MESSAGE TO NEW NODE
860

FIG. 8A

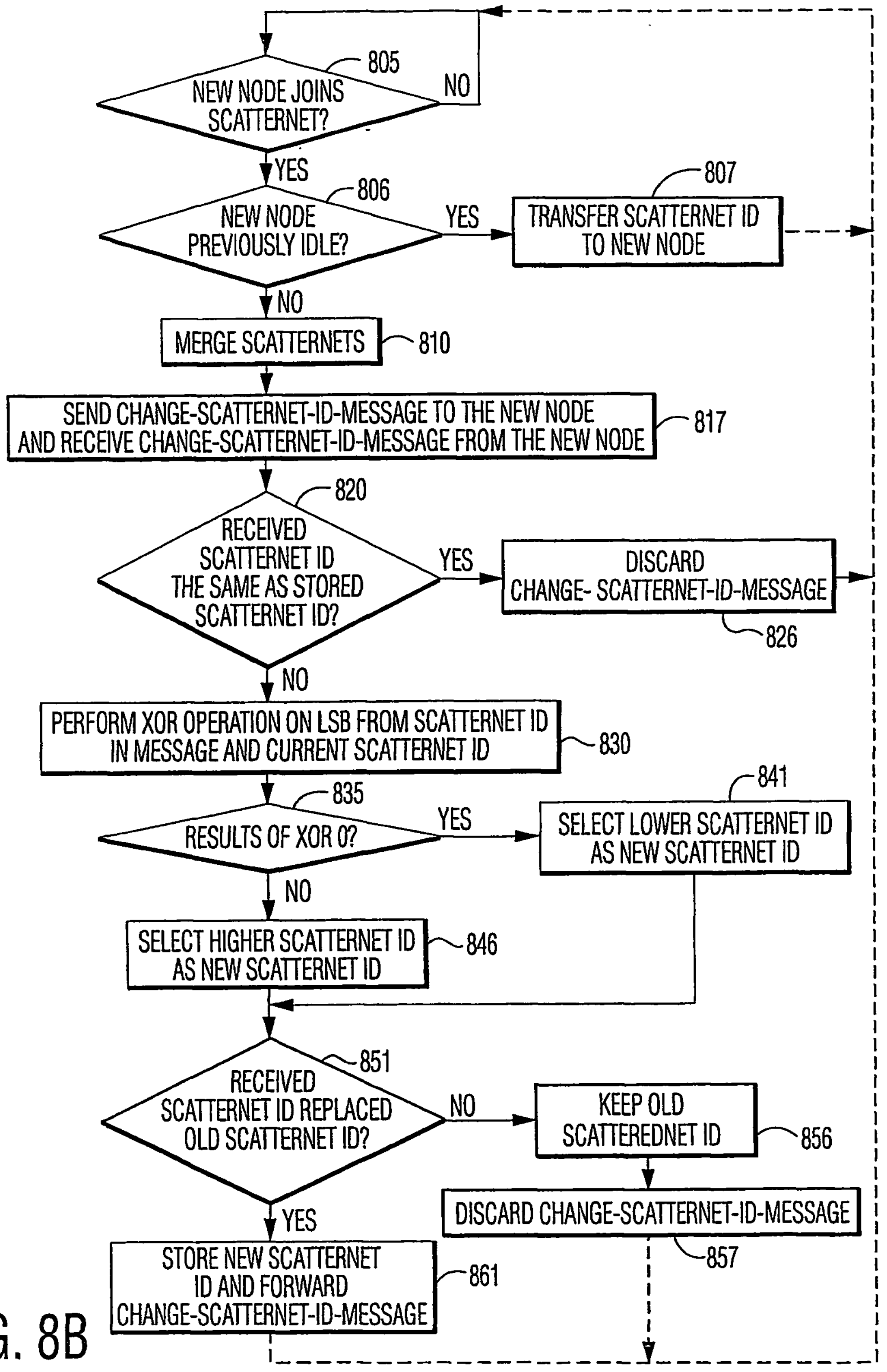

805
NEW NODE JOINS SCATTERNET?
NO
YES
806
NEW NODE PREVIOUSLY IDLE?
YES
807
TRANSFER SCATTERNET ID TO NEW NODE
NO
MERGE SCATTERNETS
810
SEND CHANGE-SCATTERNET-ID-MESSAGE TO THE NEW NODE AND RECEIVE CHANGE-SCATTERNET-ID-MESSAGE FROM THE NEW NODE
817
820
RECEIVED SCATTERNET ID THE SAME AS STORED SCATTERNET ID?
YES
DISCARD CHANGE- SCATTERNET-ID-MESSAGE
826
NO
PERFORM XOR OPERATION ON LSB FROM SCATTERNET ID IN MESSAGE AND CURRENT SCATTERNET ID
830
835
RESULTS OF XOR 0?
YES
841
SELECT LOWER SCATTERNET ID AS NEW SCATTERNET ID
NO
SELECT HIGHER SCATTERNET ID AS NEW SCATTERNET ID
846
851
RECEIVED SCATTERNET ID REPLACED OLD SCATTERNET ID?
NO
KEEP OLD SCATTEREDNET ID
856
DISCARD CHANGE-SCATTERNET-ID-MESSAGE
857
YES
STORE NEW SCATTERNET ID AND FORWARD CHANGE-SCATTERNET-ID-MESSAGE
861

FIG. 8B

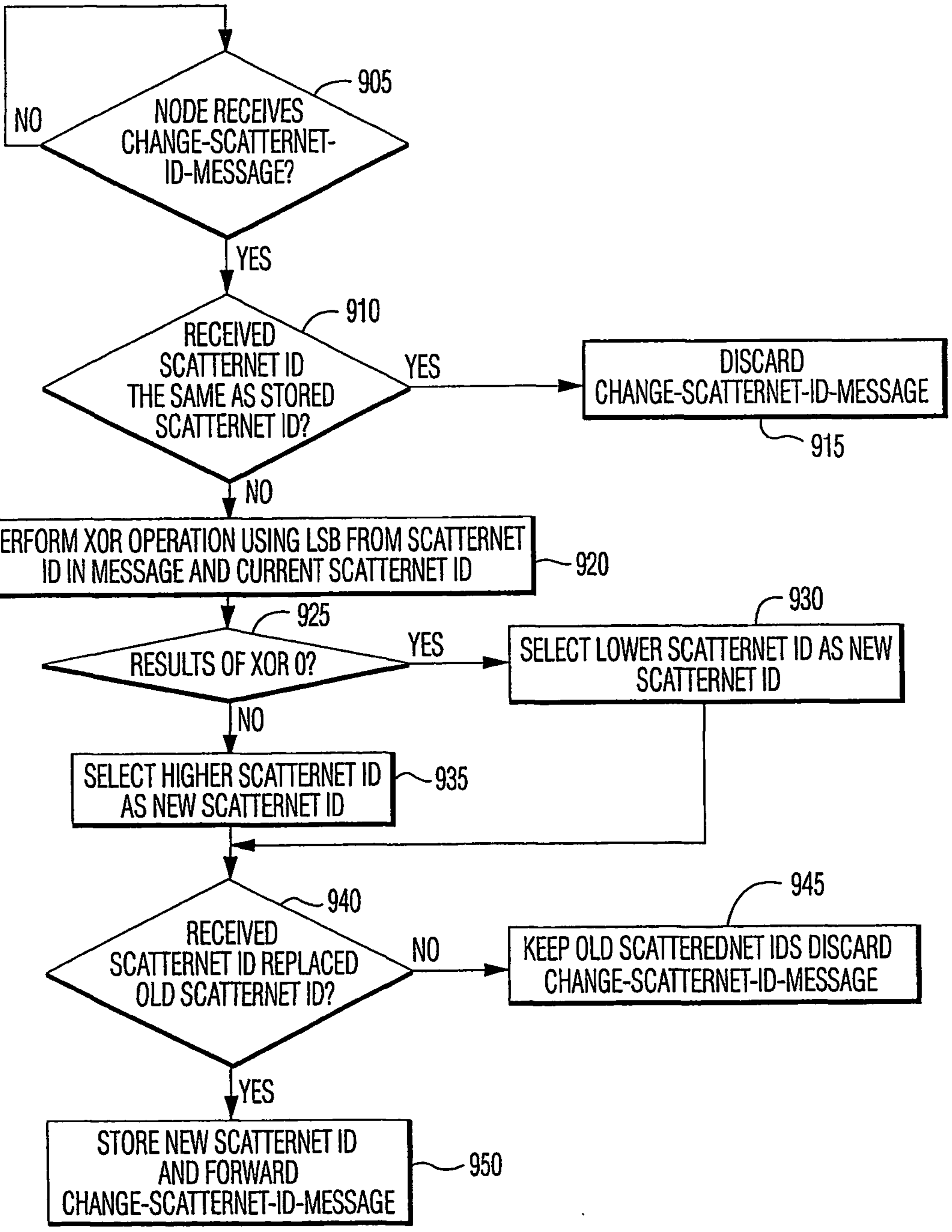

NODE RECEIVES CHANGE-SCATTERNET-ID-MESSAGE?
905
NO
YES
RECEIVED SCATTERNET ID THE SAME AS STORED SCATTERNET ID?
910
YES
DISCARD CHANGE-SCATTERNET-ID-MESSAGE
915
NO
PERFORM XOR OPERATION USING LSB FROM SCATTERNET ID IN MESSAGE AND CURRENT SCATTERNET ID
920
RESULTS OF XOR 0?
925
YES
SELECT LOWER SCATTERNET ID AS NEW SCATTERNET ID
930
NO
SELECT HIGHER SCATTERNET ID AS NEW SCATTERNET ID
935
RECEIVED SCATTERNET ID REPLACED OLD SCATTERNET ID?
940
NO
KEEP OLD SCATTEREDNET IDS DISCARD CHANGE-SCATTERNET-ID-MESSAGE
945
YES
STORE NEW SCATTERNET ID AND FORWARD CHANGE-SCATTERNET-ID-MESSAGE
950

FIG. 9

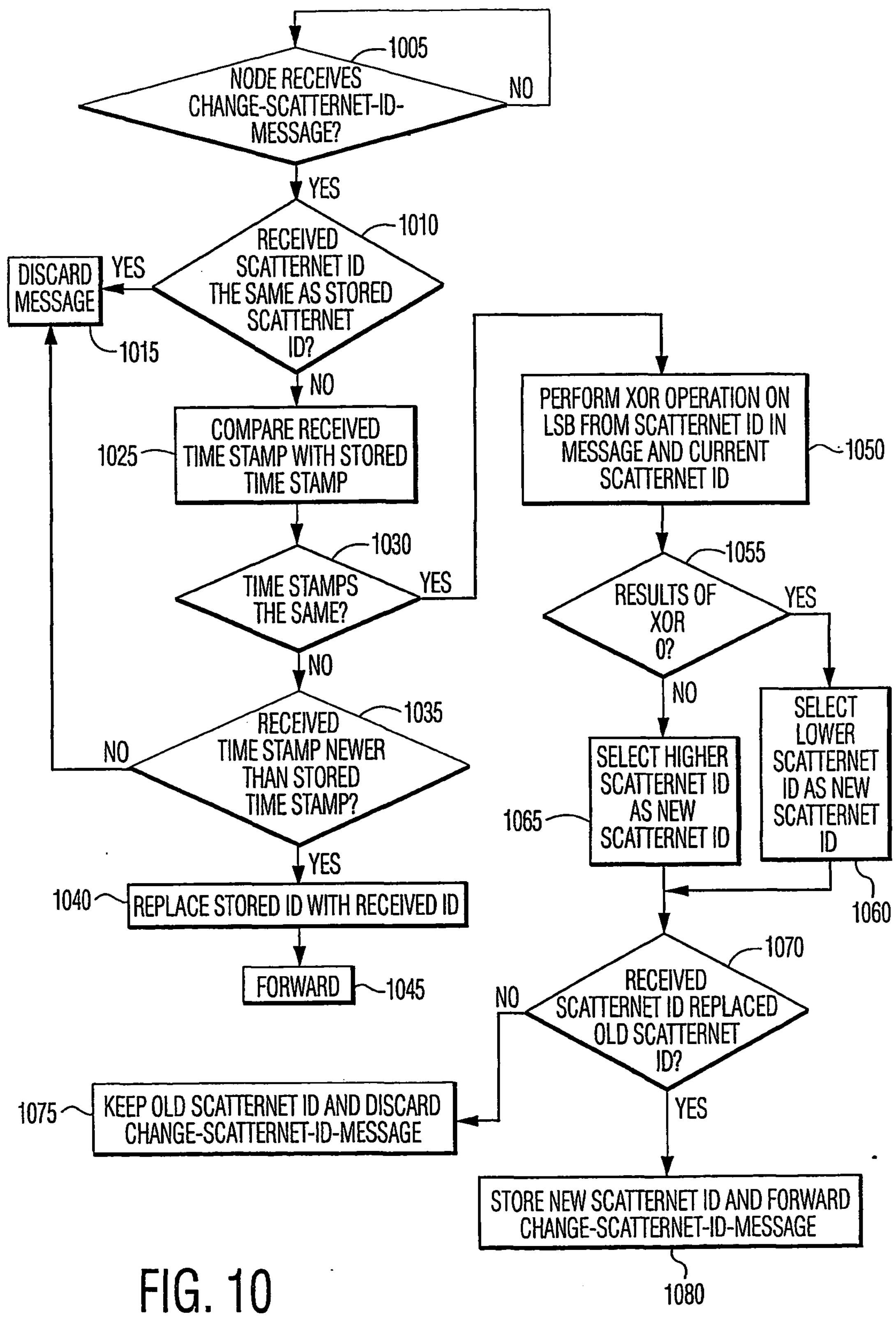

NODE RECEIVES CHANGE-SCATTERNET-ID-MESSAGE?
1005
NO
YES
RECEIVED SCATTERNET ID THE SAME AS STORED SCATTERNET ID?
1010
YES
DISCARD MESSAGE
1015
NO
COMPARE RECEIVED TIME STAMP WITH STORED TIME STAMP
1025
TIME STAMPS THE SAME?
1030
YES
NO
RECEIVED TIME STAMP NEWER THAN STORED TIME STAMP?
1035
NO
YES
REPLACE STORED ID WITH RECEIVED ID
1040
FORWARD
1045
PERFORM XOR OPERATION ON LSB FROM SCATTERNET ID IN MESSAGE AND CURRENT SCATTERNET ID
1050
RESULTS OF XOR 0?
1055
YES
NO
SELECT HIGHER SCATTERNET ID AS NEW SCATTERNET ID
1065
SELECT LOWER SCATTERNET ID AS NEW SCATTERNET ID
1060
RECEIVED SCATTERNET ID REPLACED OLD SCATTERNET ID?
1070
NO
YES
KEEP OLD SCATTERNET ID AND DISCARD CHANGE-SCATTERNET-ID-MESSAGE
1075
STORE NEW SCATTERNET ID AND FORWARD CHANGE-SCATTERNET-ID-MESSAGE
1080

FIG. 10

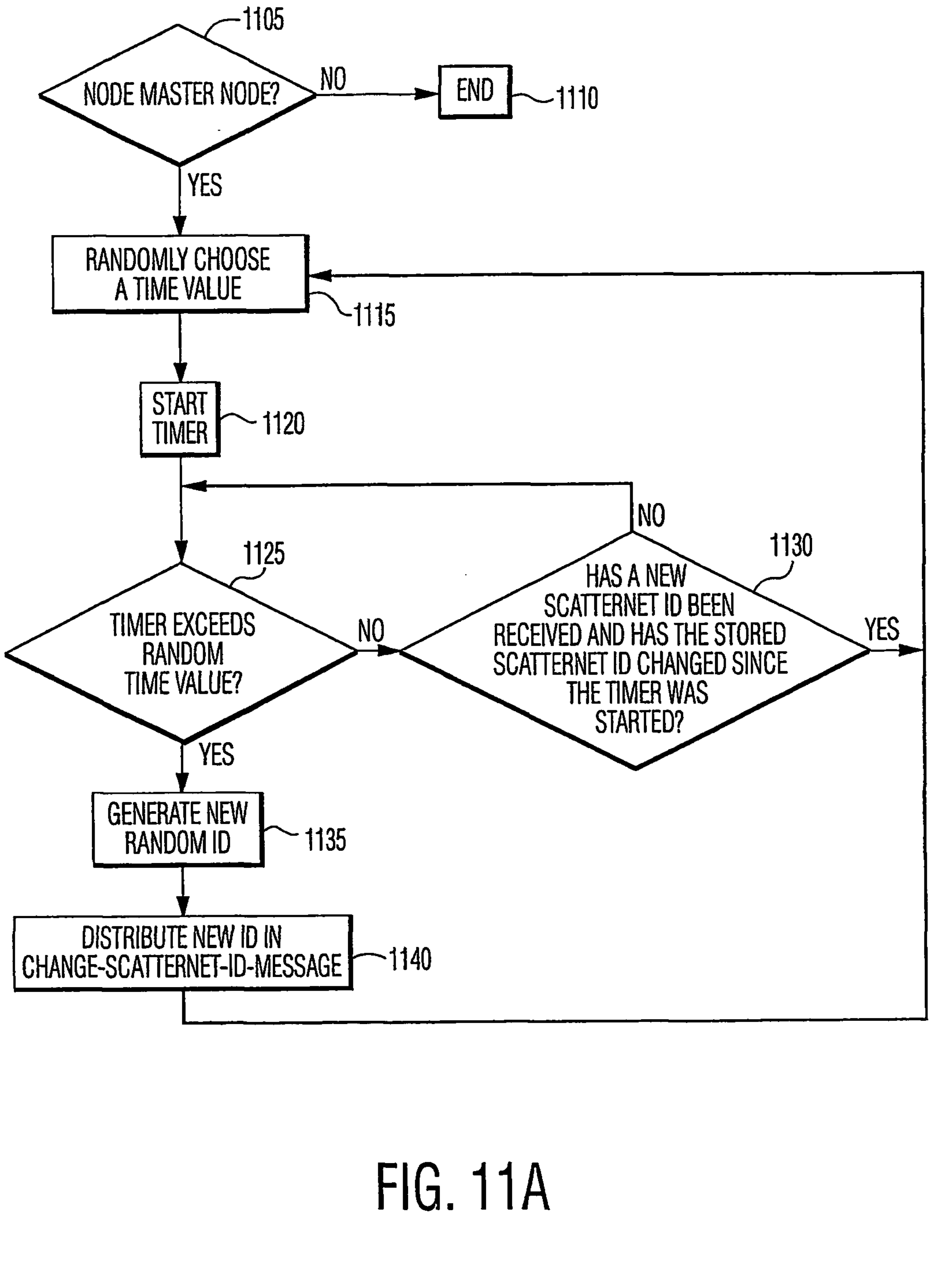
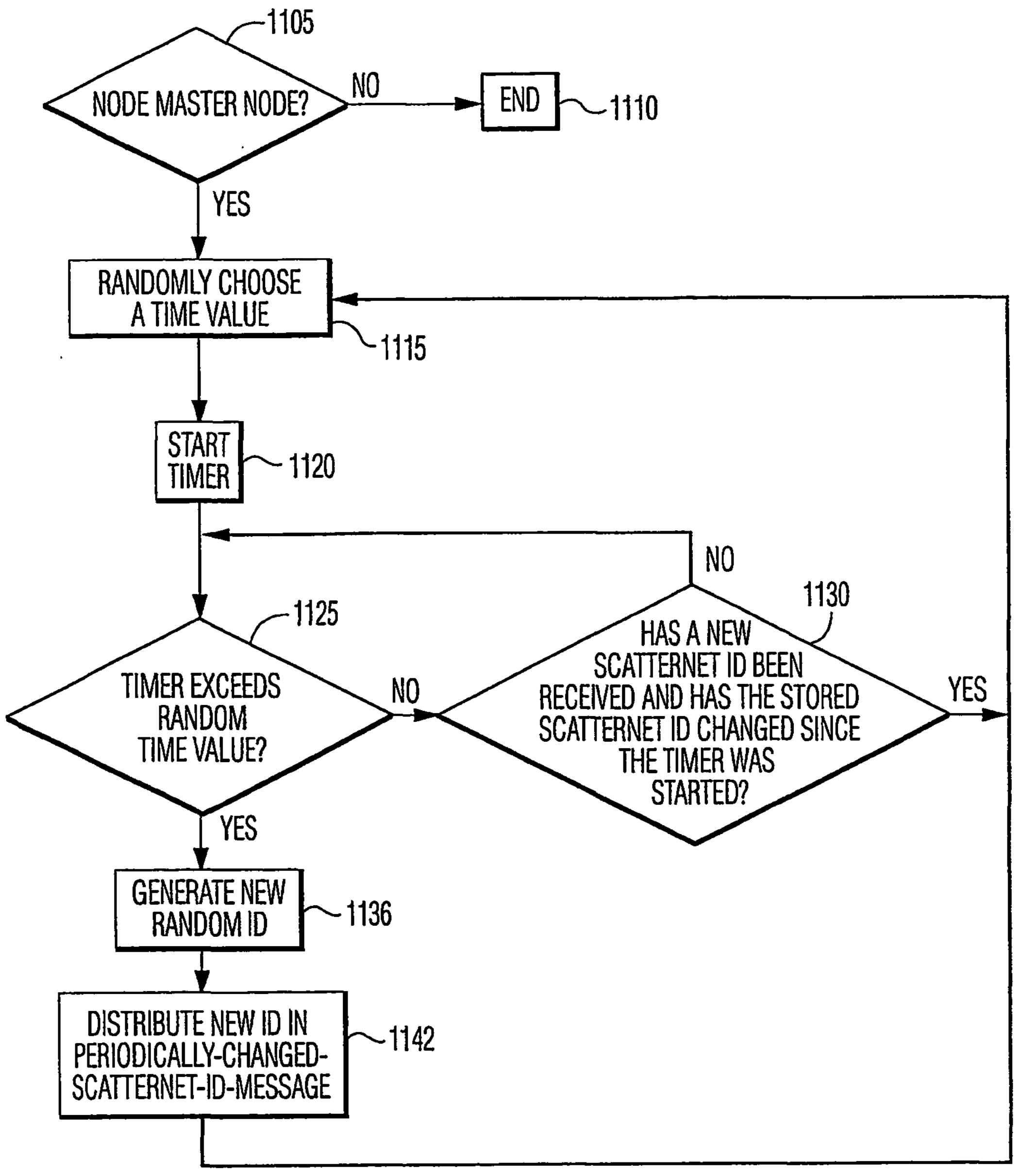

1105
NODE MASTER NODE?
NO
END
1110
YES
RANDOMLY CHOOSE A TIME VALUE
1115
START TIMER
1120
NO
1125
1130
HAS A NEW SCATTERNET ID BEEN RECEIVED AND HAS THE STORED SCATTERNET ID CHANGED SINCE THE TIMER WAS STARTED?
TIMER EXCEEDS RANDOM TIME VALUE?
NO
YES
YES
GENERATE NEW RANDOM ID
1135
DISTRIBUTE NEW ID IN CHANGE-SCATTERNET-ID-MESSAGE
1140

FIG. 11A



1105
NODE MASTER NODE?
NO
END
1110
YES
RANDOMLY CHOOSE A TIME VALUE
1115
START TIMER
1120
NO
1130
1125
HAS A NEW SCATTERNET ID BEEN RECEIVED AND HAS THE STORED SCATTERNET ID CHANGED SINCE THE TIMER WAS STARTED?
YES
TIMER EXCEEDS RANDOM TIME VALUE?
NO
YES
GENERATE NEW RANDOM ID
1136
DISTRIBUTE NEW ID IN PERIODICALLY-CHANGED-SCATTERNET-ID-MESSAGE
1142

FIG. 11B

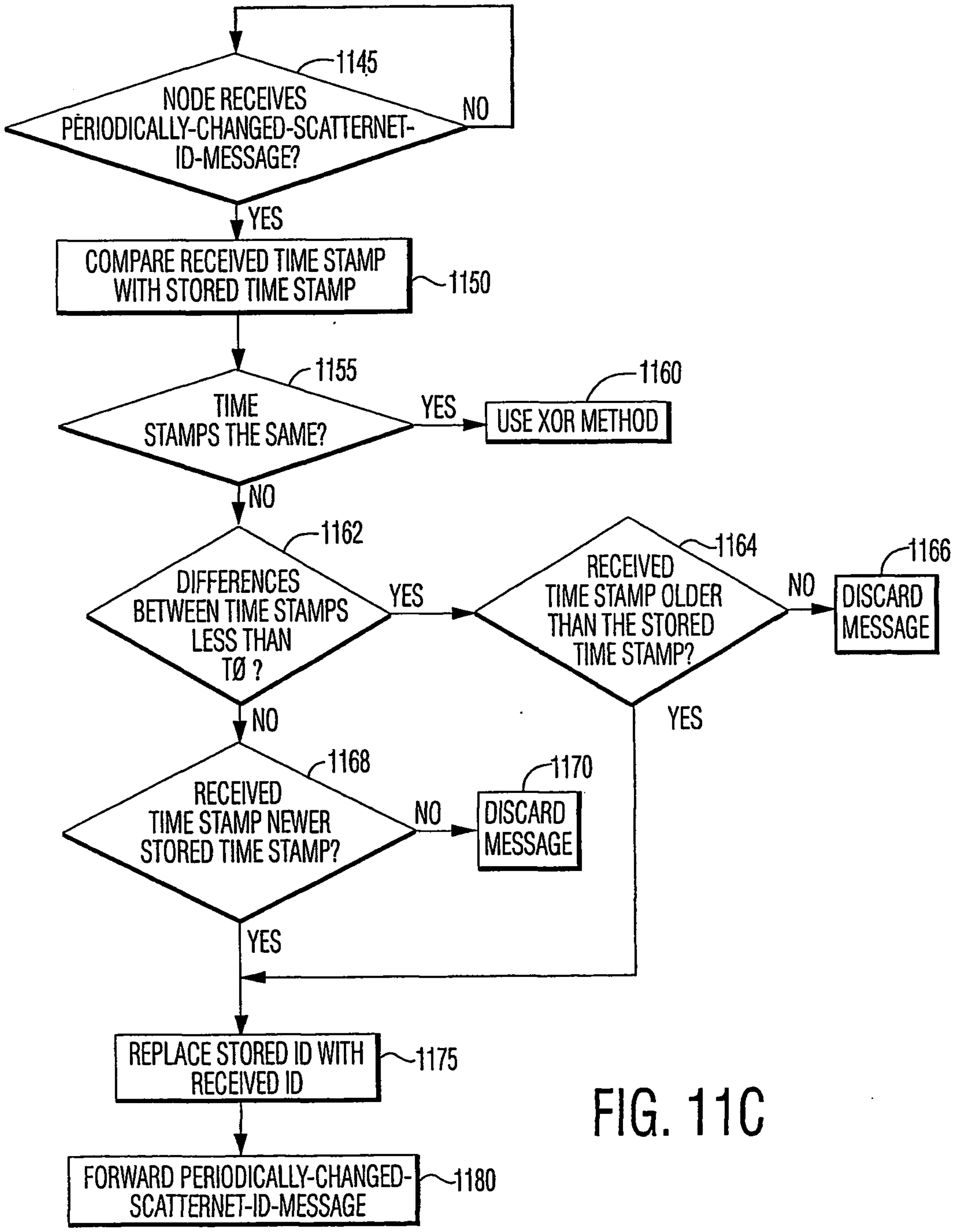

1145
NODE RECEIVES PERIODICALLY-CHANGED-SCATTERNET-ID-MESSAGE?
NO
YES
COMPARE RECEIVED TIME STAMP WITH STORED TIME STAMP
1150
1155
TIME STAMPS THE SAME?
YES
1160
USE XOR METHOD
NO
1162
DIFFERENCES BETWEEN TIME STAMPS LESS THAN TØ ?
YES
1164
RECEIVED TIME STAMP OLDER THAN THE STORED TIME STAMP?
NO
1166
DISCARD MESSAGE
YES
NO
1168
RECEIVED TIME STAMP NEWER STORED TIME STAMP?
NO
1170
DISCARD MESSAGE
YES
REPLACE STORED ID WITH RECEIVED ID
1175
FORWARD PERIODICALLY-CHANGED-SCATTERNET-ID-MESSAGE
1180

FIG. 11C

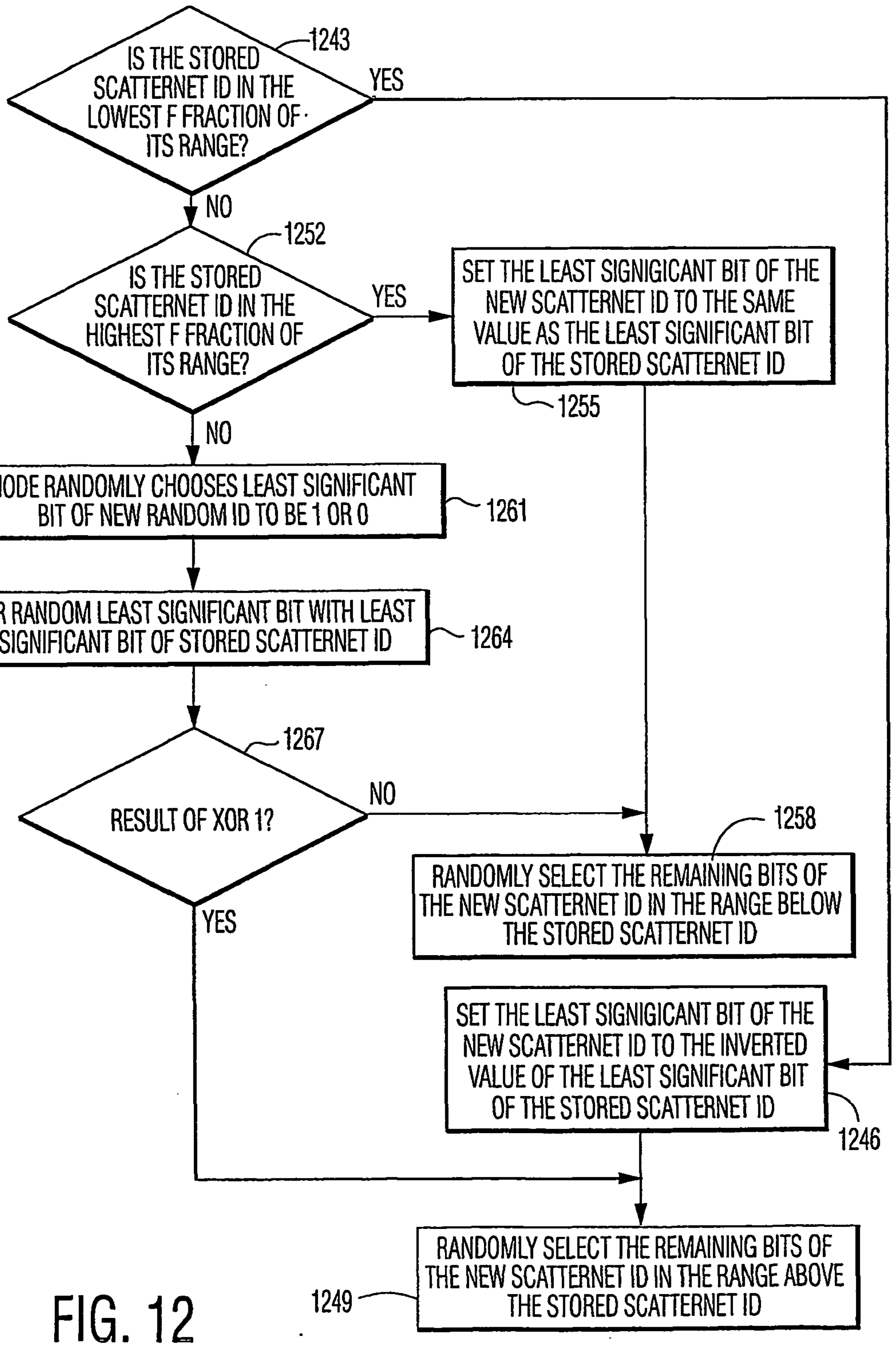

IS THE STORED SCATTERNET ID IN THE LOWEST F FRACTION OF ITS RANGE?
1243
YES
NO
IS THE STORED SCATTERNET ID IN THE HIGHEST F FRACTION OF ITS RANGE?
1252
YES
SET THE LEAST SIGNIGICANT BIT OF THE NEW SCATTERNET ID TO THE SAME VALUE AS THE LEAST SIGNIFICANT BIT OF THE STORED SCATTERNET ID
1255
NO
NODE RANDOMLY CHOOSES LEAST SIGNIFICANT BIT OF NEW RANDOM ID TO BE 1 OR 0
1261
XOR RANDOM LEAST SIGNIFICANT BIT WITH LEAST SIGNIFICANT BIT OF STORED SCATTERNET ID
1264
RESULT OF XOR 1?
1267
NO
1258
RANDOMLY SELECT THE REMAINING BITS OF THE NEW SCATTERNET ID IN THE RANGE BELOW THE STORED SCATTERNET ID
YES
SET THE LEAST SIGNIGICANT BIT OF THE NEW SCATTERNET ID TO THE INVERTED VALUE OF THE LEAST SIGNIFICANT BIT OF THE STORED SCATTERNET ID
1246
RANDOMLY SELECT THE REMAINING BITS OF THE NEW SCATTERNET ID IN THE RANGE ABOVE THE STORED SCATTERNET ID
1249

FIG. 12

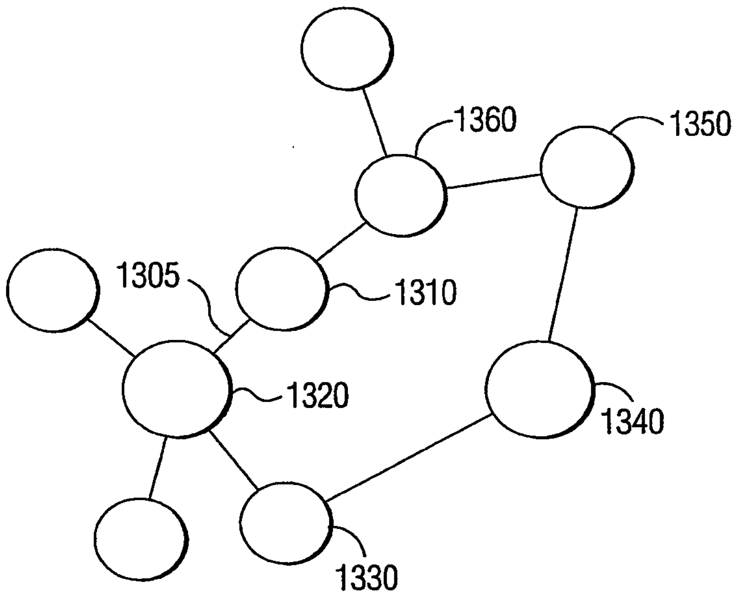


FIG. 13

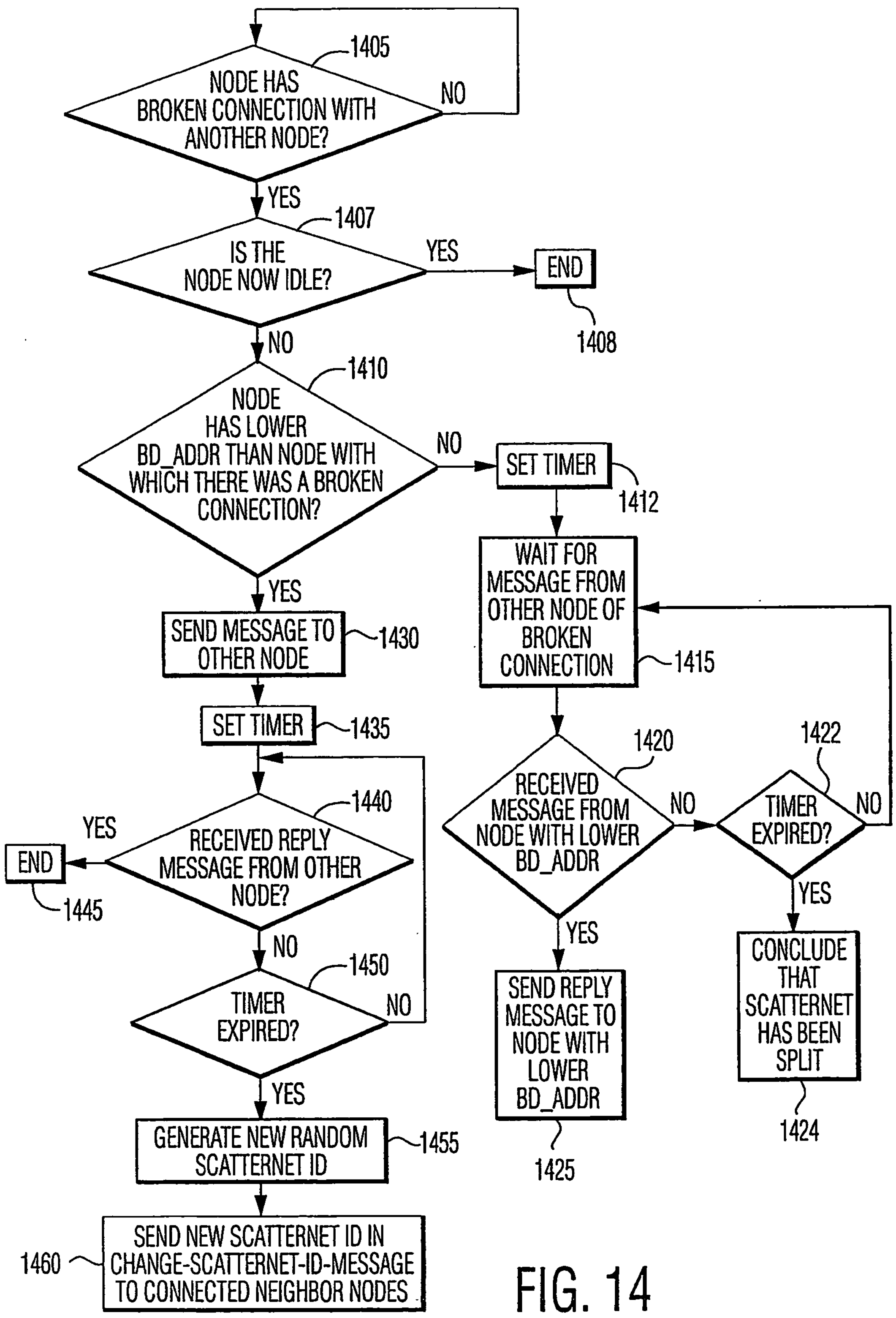

1405
NODE HAS BROKEN CONNECTION WITH ANOTHER NODE?
NO
YES
1407
IS THE NODE NOW IDLE?
YES
END
1408
NO
1410
NODE HAS LOWER BD_ADDR THAN NODE WITH WHICH THERE WAS A BROKEN CONNECTION?
NO
SET TIMER
1412
WAIT FOR MESSAGE FROM OTHER NODE OF BROKEN CONNECTION
1415
YES
SEND MESSAGE TO OTHER NODE
1430
SET TIMER
1435
1440
RECEIVED REPLY MESSAGE FROM OTHER NODE?
YES
END
1445
NO
1450
TIMER EXPIRED?
NO
YES
GENERATE NEW RANDOM SCATTERNET ID
1455
1460
SEND NEW SCATTERNET ID IN CHANGE-SCATTERNET-ID-MESSAGE TO CONNECTED NEIGHBOR NODES
1420
RECEIVED MESSAGE FROM NODE WITH LOWER BD_ADDR
NO
1422
TIMER EXPIRED?
NO
YES
SEND REPLY MESSAGE TO NODE WITH LOWER BD_ADDR
1425
YES
CONCLUDE THAT SCATTERNET HAS BEEN SPLIT
1424

FIG. 14

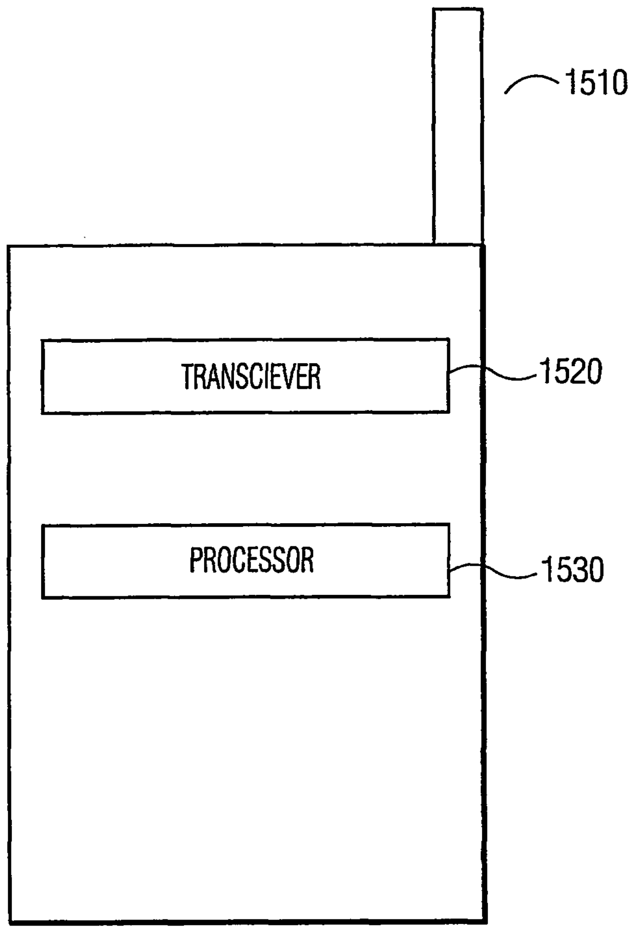


FIG. 15